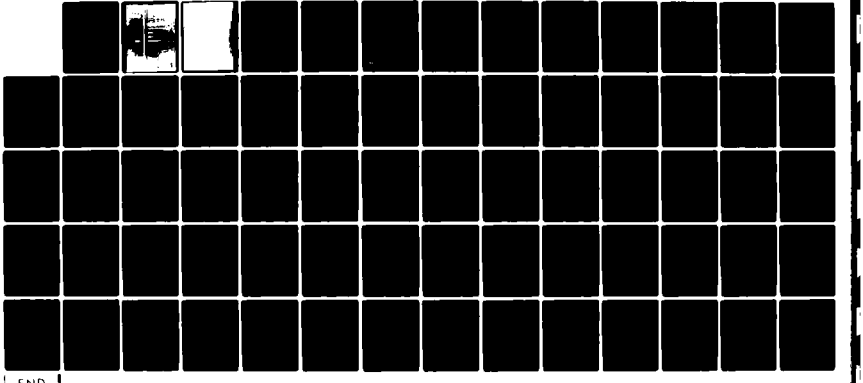
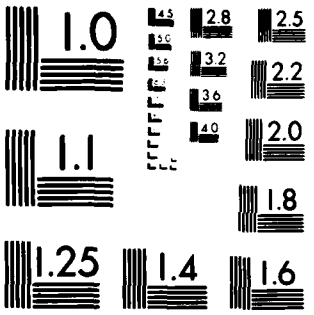
AD-A146 384 DYNA-SIM: A NONSTATIONARY QUEUING SIMULATION WITH APPLICATION TO THE AUTO..(U) RAND CORP SANTA MONICA CA L W MILLER ET AL. JUL 84 RAND/N-2087-AF 1/1

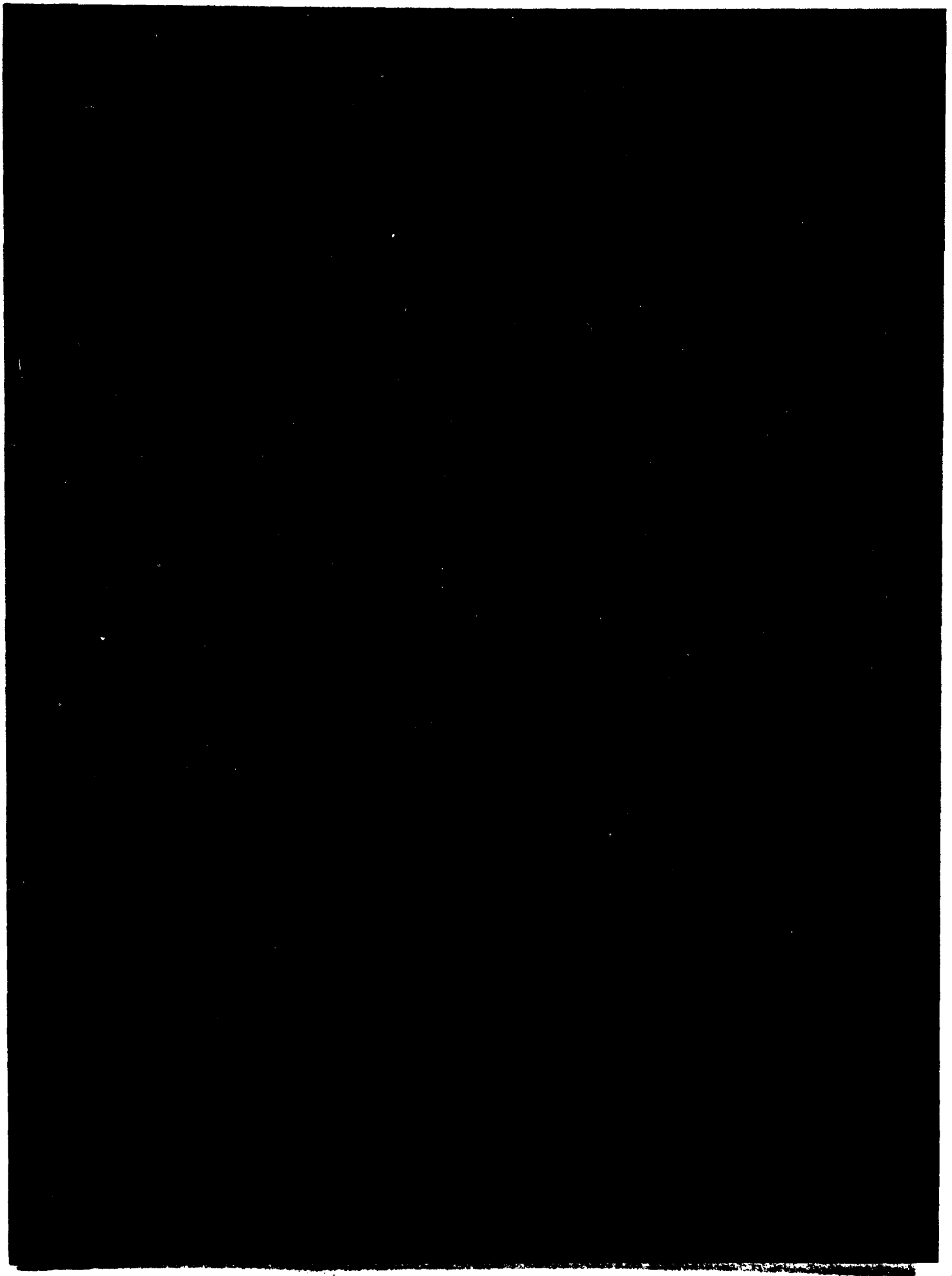
UNCLASSIFIED F49620-82-C-0018 F/G 15/5 NL

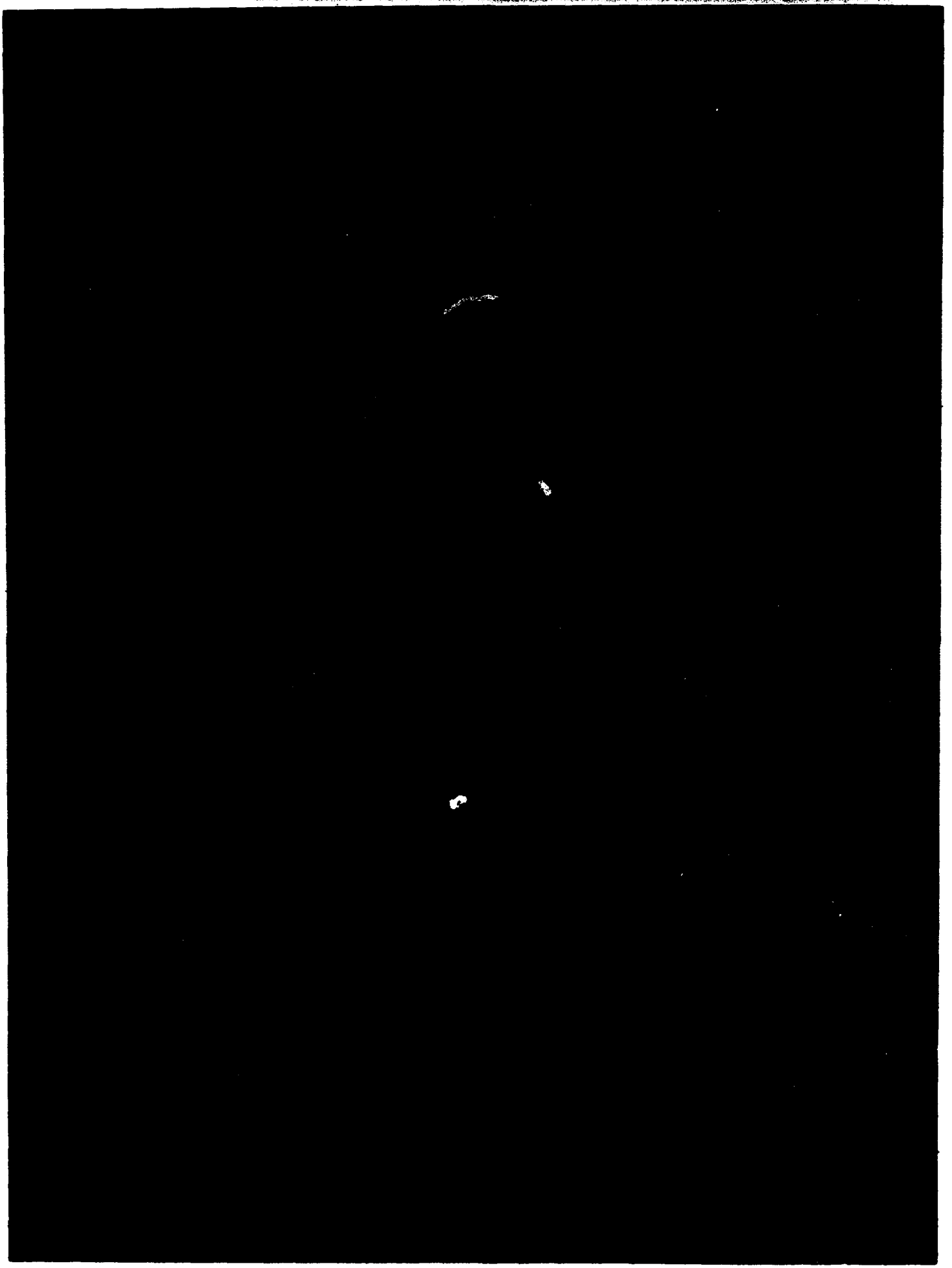


END
DATE FILMED
11-84
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A





UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER N-2087-AF	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DYNA-SIM: A Nonstationary Queuing Simulation with Application to the Automated Test Equipment Problem		5. TYPE OF REPORT & PERIOD COVERED Interim 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) L. W. Miller, R. E. Stanton, G. B. Crawford		8. CONTRACT OR GRANT NUMBER(s) F49620-82-C-0018
9. PERFORMING ORGANIZATION NAME AND ADDRESS The Rand Corporation 1700 Main Street Santa Monica, CA. 90406		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Requirements, Programs & Studies Group (AF/RDQM) Ofc, DCS/R&D and Acquisition Hq USAF, Washington, D.C. 20330		12. REPORT DATE July 1984 13. NUMBER OF PAGES 59
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release: Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

No Restrictions

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Spare parts
Maintenance
Repair
Test equipment
Simulation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See reverse side

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

This Note describes the Dyna-Sim model, which provides a capability for exploring the implications of maintenance repair queuing and requirements for spare parts. Several useful conclusions emerge from the study of the Automated Test Equipment (ATE) queuing problem using Dyna-Sim, which are summarized as follows: The ample server assumption (more than enough ATE available to serve any repair demands) is a very poor approximation when queues become saturated in high aircraft sortie rate scenarios; and if repair times follow the exponential distribution, certain approximation techniques become available (because the queuing system satisfies requirements of a Markov process). But real-world repair times are rarely exponential. The authors used Dyna-Sim to show that in a constrained server problem, the choice of the repair time distribution is not important. Thus, analytical approximations for queuing in capability assessment models are tractable.

A RAND NOTE

DYNA-SIM: A NONSTATIONARY QUEUING SIMULATION WITH APPLICATION TO THE AUTOMATED TEST EQUIPMENT PROBLEM

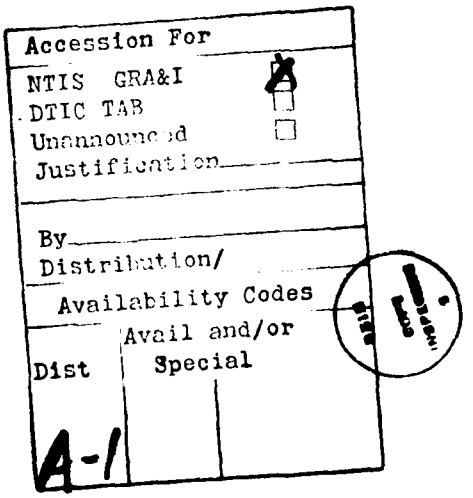
L. W. Miller, R. E. Stanton, G. B. Crawford

July 1984

N-2087-AF

Prepared for

The United States Air Force



Rand
1700 MAIN STREET
P.O. BOX 2138
SANTA MONICA, CA 90406-2138

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

PREFACE

In all recent U.S. combat aircraft, the armed services have sought the increased capability inherent in suites of digital avionics. These avionics generally rely on large and expensive automated test equipment (ATE) for repair.

In the event of a fast-moving war with high sortie rates, the removal rates for such avionics are likely to exceed the repair rates for the ATE, resulting in substantial queues. To assess the ability to deploy aircraft and "fly a scenario" with high sortie rates, one must be able to model and quantify the degree of degradation that will result from such queuing.

The model described in this Note was built at The Rand Corporation to simulate the queuing for ATE and quantify its importance in the repair cycle; it was also used to evaluate the accuracy of several analytic approximations. This research was conducted in the Project AIR FORCE Resource Management Program under the project "The Driving Inputs and Assumptions of Stockage/Assessment Models."

This Note should be of special interest to logistics analysts and resource planners.

SUMMARY

This Note describes the Dyna-Sim model, which provides a capability for exploring the implications of maintenance repair queuing and requirements for spare parts. Using Dyna-Sim, several analytical approximations to repair queuing processes related to Automated Test Equipment (ATE) were examined. There are several motivations behind this analysis:

- Repair of avionics equipment has become quite dependent on ATE, which is expensive (thus scarce), and prone to failure.
- Capability assessment models dealing with the Air Force's ability to deploy and fly a scenario with high sortie rates must be able to quantify the degradation that results from queuing of avionics awaiting repair on ATE. Almost all such models now make an "ample server" assumption that ignores both wartime and peacetime queuing. Such modeling is misleading at best and potentially dangerous.
- Several analytic approximations to the ATE queuing process have been considered at Rand for incorporating into Dyna-METRIC, the Rand-developed Air Force standard capability assessment model.
- Evaluation of these analytic approximations could best be accomplished by building a straightforward Monte Carlo discrete-event simulation model that would explicitly model ATE queuing, preferably in a fast-moving war with high aircraft sortie rates that would generate high demands on the ATE.

Several useful conclusions emerge from our study of the ATE queueing problem using Dyna-Sim, which we summarize as follows:

- **The ample server assumption (more than enough ATE available to serve any repair demands) is a very poor approximation when queues become saturated in high aircraft sortie rate scenarios.**

- If repair times follow the exponential distribution, certain approximation techniques become available (because the queuing system satisfies requirements of a Markov process). But real-world repair times are rarely exponential. **We used Dyna-Sim to show that in a constrained server problem, the choice of the repair time distribution is not important. Thus, analytical approximations for queuing in capability assessment models are tractable.**

ACKNOWLEDGMENT

The authors were fortunate to have Jack Abell review this Note. He provided a thorough and detailed review that greatly improved its organization, content, and especially readability.

CONTENTS

PREFACE iii

SUMMARY v

ACKNOWLEDGMENT vii

FIGURES xi

Section

I. INTRODUCTION 1

The Problem 1
Mathematical Queuing Models and Their Assumptions 2
Overview 4

II. THE DYNA-SIM SIMULATION MODEL 6

Simulation Time Management 6
Modeling Arrival Processes with Variance-to-Mean Ratios Greater Than 1 10
Job Service Selection Priority Rules 11
Random Number Streams 12

III. RESULTS OF SOME APPLICATIONS OF DYNA-SIM 14

Appendixes

A. INPUTS AND OUTPUTS OF THE DYNA-SIM MODEL 19
B. DYNA-SIM SIMSCRIPT II COMPUTER PROGRAM 35

BIBLIOGRAPHY 59

FIGURES

1. Dyna-Sim Time Intervals 8

2. Mean Value Function Used To Determine Time of Next Arrival .. 9

I. INTRODUCTION

THE PROBLEM

For its most recent combat aircraft, the Air Force and the other services have used avionics suites that rely on expensive automated test equipment (ATE) for repair. Several characteristics of this trend present an added challenge to the logistician concerned with wartime capability: (1) the individual avionics units are expensive and, in some important cases, in short supply; (2) they are complex and fault isolation can be difficult; (3) repair without the ATE is usually impossible at the base level; and (4) the ATE also is expensive, in short supply, and subject to failure.

ATE test equipment may operate at near capacity in peacetime, but in a wartime environment, especially in NATO, avionics repair demands may cause considerable delays in repairs because of ATE queuing. Several properties of the test equipment exacerbate the queuing problem. The equipment is expensive, in short supply, and failure prone; it is also large, heavy, and requires extensive airlift capacity. In most scenarios, aircraft are deployed from the United States much more rapidly than their supporting ATE, thus increasing the load on the already limited test equipment permanently located in Europe.

Models to estimate delays and numbers of items in repair are needed to assess capability, allocate resources, and establish policies for using ATE. In the abstract, the situation is a queuing problem, but it has complexities that make the problem more difficult than analogous problems cited in the literature. For example, the intensity of the surge and its importance preclude steady state approximations. In addition, the number of servers (stands of ATE) may change in a deterministic manner as ATE is deployed, and additionally may change at random as they fail. Perhaps most important, service (repair) priorities will not be first come first served (FCFS). Instead, actual service priorities can be expected to favor a rule in which the component causing the most downed aircraft is served next.

In many constrained repair situations, it is reasonable to believe that in wartime the limit on service capacity will be mitigated by the flexibility of manpower and efficient deviations from standard procedures. However, in the case of ATE, it is often impossible to reduce the considerable service time required on the test equipment to isolate a fault or verify that the unit has been repaired. This inflexibility makes especially inappropriate and misleading the common and mathematically convenient "ample server" assumption, which is that items never wait for repair because of limited numbers of people or equipment.

We have focused attention on the ATE queuing problem, but constrained service capability causes many other queuing problems, such as the effect of limited manpower on repair, constrained depot repair, flight line turnaround time constrained by POL facilities. The ATE queuing problem has been singled out because of its importance and the relative inflexibility of the ATE itself.

The ample server assumption has pervaded both previous and current generations of Air Force logistics capability assessment models. In particular, this assumption is used in early versions of the Dyna-METRIC model[1] and similar AF models (such as WARS) now under development. The invalidity of the ample server assumption in these models motivated the development of Dyna-Sim and the companion analytic models as research tools to provide alternative approaches to that assumption. Current versions of the Dyna-METRIC models at Rand now incorporate a version of the analytic model.

MATHEMATICAL QUEUING MODELS AND THEIR ASSUMPTIONS

Where arrival intensities are zero and then jump up to a constant level, *and* we can make the infinite or ample server assumption, the closed form solution for the number of parts in the system is well known.[2] Although these assumptions are unduly restrictive, the closed

[1]R. J. Hillestad and M. J. Carrillo, "Models and Techniques for Recoverable Item Stockage When Demands and the Repair Process are Nonstationary--Part I: Performance Measurement," The Rand Corporation, N-1482-AF, November 1980.

[2]L. Takacs, *Introduction to the Theory of Queues*, Oxford University Press, New York, 1962.

form solution provides a valuable benchmark to check the accuracy of more complex models.

If we can make the assumption that repair times are exponential random variables, then the performance of the queue can be approximated by a process where the number of line replaceable units (LRUs) in repair is a Markov process. This is important because the probability distributions associated with a Markov process can be expressed as the solution to a certain set of differential equations. These Kolmogoroff equations are well suited to numerical solution on a computer. A computer program for the solution of these equations has been developed and used at Rand.

Dyna-Sim is a Monte Carlo simulation of a multi-server, multi-job-class queuing system that allows variation of the arrival rates for the classes of jobs be varied over time. It is able to vary arrival rates, making it different from other queuing system simulations. Jobs are selected from the queue according to a priority rule chosen by the user. Four alternative priority rules have been implemented; others should not be difficult to add. Jobs arrive according to a time-varying Poisson or compound Poisson process, and the user may choose either exponentially distributed or constant service times. Other service time distributions could be easily included.

Although analytic models are generally preferred over simulations, it does not take much in the way of complexity to frustrate attempts at constructing analytic models. In fact, the original motivation for constructing the Dyna-Sim model was to provide a way to test the adequacy of analytic models based on solving differential equations. This investigation of queuing models, both analytic and the simulation approach described here, was motivated by the need to model the effects of delays in the repair of components induced by the constrained capacity of ATE, which is a limited resource that must be shared among several types of items.

When the sum of the arrival rates exceeds the sum of the service rates we say the queuing system is saturated. In this case the queue is highly variable, and achieving a given degree of accuracy with any such Monte Carlo model may require many trial runs.

The ATE queuing problem can be characterized by:

1. A collection of different incoming part classes requiring service, LRU(i), i = 1,2,...,N,
2. Each having a mean service time T(i),
3. A time-varying arrival intensity H(i,t),
4. An initial level of serviceable stock S(i),
5. A fixed number of servers NS.

Let N(i,t) be the random number of units of LRU(i) in the queue and in repair at time t.

Throughout we assume that parts fail and arrive at the model's single queue according to a Poisson arrival process with a nonstationary arrival intensity.[3]

Several real world service priority schemes are suggested by the nature of the queuing problem. We have included four options in Dyna-Sim. Rule 1 invokes FCFS service priority. Rule 2 assigns priority to the class of jobs with the most in the queue. Rule 3 assumes that job classes have been prioritized in advance and service is given on a FCFS basis within the job class in the queue having the highest priority. In Rule 4 we compute the back order quantity BOQ(i) = N(i,t) - S(i), and service is given to the job class with the highest back order quantity.

OVERVIEW

The remainder of the Note describes the Dyna-Sim model and presents some results obtained with it. Section II discusses the structure of the model including discrete event simulation time management, sampling interarrival times, job service selection priority rules, and random number sampling. Section III presents some results obtained with the model. Appendix A describes each of the Dyna-Sim model input requirements, organized into seven input sets, followed by examples of outputs produced by the model. Appendix B discusses the Simscript II program, its variables, subprograms, and event control.

---

[3] For a detailed discussion of Poisson and compound Poisson arrival processes see G. B. Crawford, *Palm's Theorem for Nonstationary Processes*, The Rand Corporation, R-2750-RC, October 1981.

It presents some guidelines for adding service priority rules and a listing of the Simscript II code.

II. THE DYNA-SIM SIMULATION MODEL

Dyna-Sim is a Monte Carlo discrete event simulation of a multi-server, multi-job-class queuing system with time-varying arrival rates for each job class. Built into the model are the four job service selection rules mentioned earlier; others can be added with minimal difficulty. In this section, we first present an overview of the management of simulated time and then discuss its relation to statistical data collection. Four additional topics receive detailed attention: sampling interarrival times, modeling arrival processes when the variance exceeds the mean, job service selection priority rules, and the handling of random number streams.

SIMULATION TIME MANAGEMENT

Most queuing simulations are done with the intention of studying the steady-state behavior of a system with time-invariant parameters. The essence of Dyna-Sim, however, is to study behavior over a finite time period during which arrival rates vary as specified by the user. Thus, managing simulated time is somewhat complicated by two things. The first is the arrangement for varying arrival rates, and the second is the need to run several trials to obtain statistically reliable results. Dyna-Sim deals with four kinds of time intervals: trial, run-in, period, and sample.

A trial is a time interval during which we desire to study the system's behavior. To get meaningful results, the analyst would want statistics computed from several trials. One run of the model will provide data from an arbitrary number of trials to be specified by the user. Before each trial, the system is allowed to run for an interval called a run-in, during which arrival rates are held constant. The purposes of the run-in intervals are (1) to allow remaining jobs from the preceding trial to clear out of the system so that flow time (time in system) statistics can be collected on all the jobs that arrived during the preceding trial, and (2) to let the transients from the preceding trial die out, allowing the system to reach a desired start-

of-trial condition. Because of the dual purpose of run-ins, a simulation run begins with a run-in, alternates between trials and run-ins, and finishes with a final run-in. The final run-in is to capture flow time statistics on jobs remaining in the system at the end of the last trial.

In our use, we frequently wanted to have trials start with very few jobs in the system. This can be achieved by setting the arrival rate during run-in intervals to a very low value (e.g., 10^{-5}) and making the run-in period fairly long. With low arrival rates, even long run-in intervals take little computer time because of the nature of the "next imminent event" timing mechanism. The program also allows the user to specify an optional "initial run-in" interval that takes place before the first standard run-in to give an extra amount of time for stabilization at the start.

An individual trial is broken up into smaller intervals, called "periods," to represent the time varying arrival rates. A period is a time interval over which the arrival rate parameters are held constant for all job classes. The model treats the number of periods to be one more than the number of intervals during a trial. The extra period is associated with the end of a trial, and the arrival rates specified for the extra period are for use during run-ins. Input data permit definition of the number of periods (including the extra one), the time relative to the start of a trial that each period begins, and the arrival rates by job class and period.

For example, suppose the trials are to last 80 days with four arrival rates in a trial. Then the number of periods is five. We must specify five times for the periods (when they start relative to the beginning of a trial). The first must occur at time 0 and the last must occur at time 80. The time of the last period also informs the model of the total length of a trial.

One other type of time interval is involved. The model collects primary data on the number of jobs in the system by job class and overall. We obtain these statistics by observing the state of the system at the beginning of each trial and thereafter at evenly spaced intervals within trials. These are called "sample intervals." The model performs no sampling during run-ins. Observing the number of jobs

in the system at the beginning of trials can be useful in judging the adequacy of the run-in interval length. Figure 1 illustrates the time interval relationships discussed above.

The method of sampling arrival times is based on a correspondence between a time-varying Poisson process and a stationary Poisson process with intensity equal to 1. The correspondence is as follows: Let m(t) be the intensity of the nonhomogeneous process and L(t) be the mean value function, which is equal to the integral

$$\int_0^t m(x)dx$$

If arrivals in the nonhomogeneous process occur at times t_0, t_1, t_2, ..., the numbers $L(t_0)$, $L(t_1)$, $L(t_2)$,... correspond to the arrival times in a stationary Poisson process. Thus the random variables $(L(t_n) - L(t_{n-1}))$, n = 1,2..., correspond to the interarrival times in a stationary Poisson process and are exponentially distributed with mean 1. This fact is used in the model the other way around--we generate a series of exponentially distributed random variables with mean 1 corresponding to the interarrival times of a stationary process and use

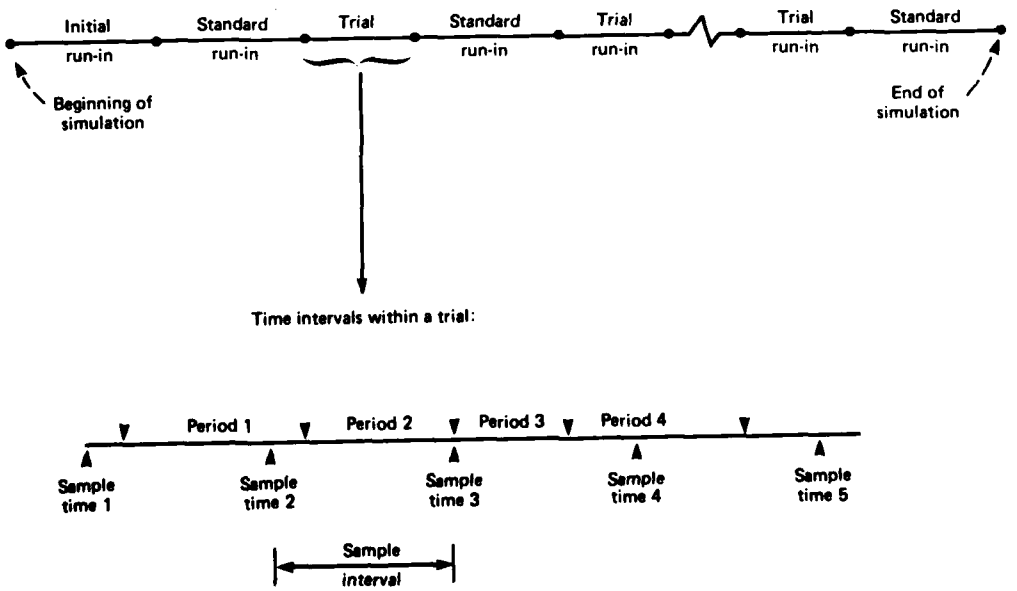


Fig. 1 — Dyna-Sim time intervals

the "functional inverse" of L to determine a sequence of arrival times for the nonstationary process.

In the model, m(t) is constant over finite intervals, which may be expressed by writing $m(t) = m_j$ for $b_{j-1} \leq t < b_j$. Note that L(t) is a nondecreasing piecewise linear function. Figure 2 portrays the relationship involved in determining arrival times. The horizontal axis is time and the vertical axis is L(t). Suppose that an event occurs at time t_{n-1}, which is ending at period b_j. The distance labeled Z on the vertical scale is the exponentially distributed random variable. To determine t_n, search for the smallest k such that $L(b_{j+k}) > L(t_{n-1}) + Z$. If $k = 0$, t_n will be in the same period as is t_{n-1}, and $t_n = t_{n-i} + (Z - L(t_{n-1}))/m$. If $k > 0$, then $t_n = b_{j+k-1}$ (for the situation shown in Fig. 2, $k = 3$).

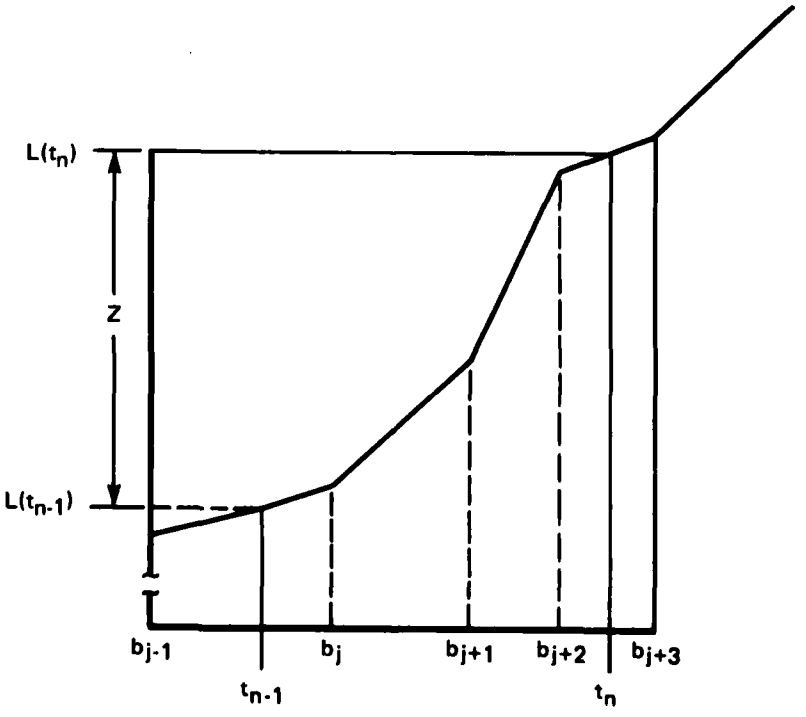


Fig. 2 — Mean value function used to determine time of next arrival

The method we have described is embodied in the routine named TOFARR (for time of arrival) that is called from the arrival event. TOFARR samples an exponential random variable and then figures out where current simulated time (TIME.V) is in relation to the cycle of periods and run-ins. In terms of the discussion above, TOFARR determines b_j and m_j. From there, it searches for the period or run-in in which the next arrival will occur. Finally, TOFARR computes the value of t_n and returns it to the arrival event routine so that the next event may be scheduled.

MODELING ARRIVAL PROCESSES WITH VARIANCE-TO-MEAN RATIOS GREATER THAN 1

For a Poisson arrival process, the distribution of the number of arrivals in a fixed time interval has a variance equal to the mean. However, actual arrival processes have variances higher than the mean. Dyna-Sim allows such processes to be modeled based on the approach adopted in METRIC.[1]

For variance-to-mean ratios greater than 1, Dyna-Sim simulates a compound Poisson process with a logarithmic compounding distribution as in METRIC. Suppose that it is desired to have an arrival process with an overall arrival rate of m jobs per time unit and a variance-to-mean ratio equal to q. For q = 1, the arrival process is a simple Poisson process with intensity equal to m. When q > 1, the desired effect is achieved by simulating a Poisson process with a lower intensity, and whenever an event occurs in the Poisson process, determine the number of arriving jobs by sampling from the compounding distribution. One can see how this scheme increases variance--times between arrival events are increased, but when arrivals occur, they happen in bunches.

The logarithmic compounding distribution has the probability mass function

$$f_x = \frac{1}{\ln q}\left(\frac{q-1}{q}\right)^x \frac{1}{x} \quad \text{for } x = 1,2,\ldots; \quad q > 1$$

[1] Craig C. Sherbrooke, *A Management Perspective on METRIC--Multi-Echelon Technique for Recoverable Item Control*, The Rand Corporation, RM-5078/1-AF, January 1968.

Notice that the compounding distribution depends only on q. The adjustment to the intensity is made by calculating

$$m_q = m\left(\frac{\ln q}{q-1}\right)$$

In Dyna-Sim, m is a function of job class and time, but q is constant for all classes and all time. Hence, only one value of q is specified by the user. Even in METRIC only one value is used for all items (because of data problems, not because of difficulties in modeling).

Within the Dyna-Sim program, the compounding scheme is invoked only if the user specifies the variance-to-mean ratio to be greater than 1. If q is less than or equal to 1, the simple Poisson processes described above are simulated.

JOB SERVICE SELECTION PRIORITY RULES

There are four job selection rules from which a user may choose by specifying the appropriate number. The selection (or dispatching) rule in use is applied every time a job is completed if there are jobs waiting. The rules are:

Rule 1 : First come, first served, with no priorities given to any job class.

Rule 2 : Select a job from the class with the most waiting jobs. First come, first serve breaks any ties within the selected class.

Rule 3 : Nonpreemptive priorities--job classes are assigned priorities and selection is made from the highest priority class with waiting jobs. Several job classes may be assigned to a single priority class.

Rule 4 : Select a job from the class with the highest backorder quantity. Backorders are in-service (repair) jobs plus jobs in queue minus stock level for a job class, which may be negative. First come, first serve is used as a tie-breaker.

Rule 4 is suggested by the ATE equipment problem described in Sec. I where a lack of serviceable items may cause grounded aircraft or other important, inoperable end items. In this case it makes sense to attempt to minimize the number of end items out of commission by choosing jobs according to the backorder status. It is assumed that the repair/supply facility starts life with an initial stock level S(i), $S(i) \geq 0$, for each item type i (or job class i). If N(t,i) is the number of items in job class i in repair plus the number in the queue at time t, then N(t,i) - S(i), the backorder quantity for item type i, is a measure of the number of end items that are out of commission for lack of item type i. If reparable units can be cannibalized then all of the holes left by missing parts can be consolidated in a minimum number of end items. This minimum number of end items out of commission will be given by the maximum of N(t,i) - S(i) over all item types.

RANDOM NUMBER STREAMS

Each job class requires a pair of random number seeds specified as input by the user. One of the seeds initiates a random number stream that is translated to successive exponential interarrival times. The other seed generates a random number stream that is translated to successive exponential service times (if stochastic, nonconstant service times are in use).

Specifying separate streams of random numbers for the arrival and service process allows the analyst to use a variance reduction technique known as "seed switching." By making runs in pairs, and reversing the order of the paired seeds (for three job classes replace the random number seeds a, b, c, d, e, f by b, a, d, c, f, e) the outcomes of the second run will be negatively correlated with the outcomes of the first run. For example, if in the first run the particular choice of seeds happened to produce statistically long interarrival times and short service times (with the result that queue sizes are small) reversing pairs of seeds will give a run with statistically short interarrival times and long service times (with the result that queue sizes are large). The average of the two runs will have less variance than a single run whose length is equivalent to the total length of the two runs.

Seed switching is not important if the only comparisons to be made are among runs of the model. If, however, one desires to compare results from the simulation with data from other sources or a different model, then seed switching is a valuable tactic. (The program displayed in Appendix B does not automatically do seed swapping; that is left to the user to do manually by making two separate runs.)

III. RESULTS OF SOME APPLICATIONS OF DYNA-SIM

As we began using Dyna-Sim several results became apparent. As expected, the infinite (or ample) server assumption is a very bad approximation to a saturated queuing problem. Several other results, although not counterintuitive with hindsight, were unexpected.

One of our concerns with the constrained queuing problem was the possible sensitivity of the results to the choice of a repair time distribution having a given mean. In particular, when striving for analytic results it is helpful to assume that service times are exponential. With this assumption a queuing system satisfies the requirements of a Markov process, and that gives the analyst a large arsenal of tools.[1] Unfortunately, real world repair times are rarely exponential. Moreover, in the infinite server case the choice of a repair time distribution having a given mean may greatly affect the distribution of the number of units in repair.[2]

Fortunately, in this more difficult constrained server problem, the repair time mean is important, but the choice of the distribution having this mean does not seem to matter when the queue is saturated. To illustrate this comparison, we used the Dyna-Sim model and compared the output of runs where (1) the repair time for LRU(i) was always equal to T(i) and (2) the repair time for LRU(i) was an exponential random variable with mean T(i). These two distributions have been picked because they are commonly used and very different.

The scenario for this comparison uses arrival rates and repair times that could be representative of a deployed wing of tactical fighters with two complete and operational suites of test equipment (the two servers). The scenario generates time-varying demands on repair that could be expected in a wartime environment. We modeled five different LRUs that get tested on one stand of the ATE suite. The

---

[1] Unpublished Rand research by Gordon B. Crawford.

[2] G. B. Crawford, *Palm's Theorem for Nonstationary Processes,* The Rand Corporation, R-2750-RC, October 1981.

maximum backorder quantities computed were maximum over these five LRUs. Rule 4, the maximum BOQ, was used to determine service priorities.

In this scenario flying increases to a rate about three or four times higher than normal peacetime flying on day 5 and then drops to a rate of about twice the peacetime rate on day 12. The queuing system becomes saturated because the arrival rate exceeds the service rate beginning on day 5 and then throughout the remainder of the scenario.

The results, rounded to two decimal places, are given in Table 1. In three cases the constant service times yielded a slightly increased performance. In three other cases the constant service times did worse. In the remaining cases the results were the same to two decimal places. These results suggest that for a fixed mean repair time the performance of the queuing system is fairly insensitive to the the choice of repair time distribution.

Table 1

EXPONENTIAL VS. CONSTANT REPAIR TIMES

	Probability of Max BOQ Not Exceeding N					
	Day 10		Day 15		Day 20	
N	exp	con	exp	con	exp	con
6	.97	.98	.78	.87	.53	.53
4	.86	.88	.55	.62	.31	.24
2	.60	.63	.25	.23	.09	.04

NOTE: exp = exponential, con = constant repair times

The other surprising result found with Rule 4 was the strong dependence between the number of backorders for different LRUs. The number of inoperable end items at time t is given by the maximum over i of $BOQ(i) = N(i,t) - S(i)$. In most models where it is desired to compute

$$\Pr[\max BOQ(i) \leq n] = \Pr[BOQ(i) \leq n \text{ for all } i] \qquad (1)$$

it is assumed that the items are independent and

$$\Pr[\text{BOQ}(i) \leq n \text{ for all } i] = \pi \Pr[\text{BOQ}(i) \leq n]. \qquad (2)$$

The latter terms are then individually computed based on the assumption that N(i,t) are Poisson random variables with known mean M(i,t).

The assumption that N(i,t) is Poisson does not seem to be a large source of error, but the assumption that the number of part i in the queue is independent of the number of part j in the queue may introduce large errors. The difference between the left and right sides of Eq.(2) is illustrated in Table 2.

Table 2

OBSERVED PROBABILITIES VS.
COMPUTATIONS BASED ON INDEPENDENCE

	Probability of Max BOQ Not Exceeding N					
	Day 10		Day 15		Day 20	
N	obs	ind	obs	ind	obs	ind
6	.97	.97	.78	.58	.53	.14
4	.86	.82	.55	.21	.31	.02
2	.60	.43	.25	.02	.09	.00

NOTE: obs = observed Dyna-Sim probabilities,
ind = probabilities calculated on the assumption that N(i,t) is independent of N(j,t).

The terms in the column headed "obs" are taken directly from the Dyna-Sim output for the scenario described above. The terms in the column headed "ind" have been computed on the assumption that N(i,t) has a Poisson distribution with the same mean used in the Dyna-Sim run, and that the number of part i in the system is independent of the number of part j. As the queue becomes saturated and the backorder quantities begin to get large, the difference between the observed distribution of max BOQ and the distribution under the independence assumption becomes

quite striking. In this case the independence assumption is very conservative because it overpredicts the number of inoperable end items. A solution to this problem will be the subject of further research.

Appendix A

INPUTS AND OUTPUTS OF THE DYNA-SIM MODEL

The Dyna-Sim computer program was developed to facilitate the analysis of various service priority rules in a multi-server, multi-job-class queuing system with time-varying arrival rates of jobs. The data and input requirements of the four service rules that we have implemented here, and the relative ease with which other rules could be introduced by other analysts, led us to categorize all model inputs into one of seven "input sets." Below we discuss the requirements of each of these input sets and indicate their applicability or inapplicability to each of the four rules. Then we present an example of the input requirements for each of the four rules. The appendix concludes with a description of the various model outputs, with examples.

DYNA-SIM MODEL INPUTS

The Dyna-Sim model, depending on the service priority rule selected, will require fully specified input data from at least five of seven possible "input sets," as we have designated our input data groups. The table below provides an overview of the content of each input set and which priority rules require data in that set.

Input		Service Rule			
Set	Description	1	2	3	4
1	System and simulation defining parameters	yes	yes	yes	yes
2	Service time parameters	yes	yes	yes	yes
3	Times for changing arrival rates in trial	yes	yes	yes	yes
4	Arrival rates for each class, period	yes	yes	yes	yes
5	Random number seeds	yes	yes	yes	yes
6	Priority level of each job class	no	no	yes	no
7	Inventory stock levels for each job class	no	no	no	yes

All time-related inputs used by Dyna-Sim should be specified consistently. In the example runs below, we describe cases with time units measured in days; however the user may work with any time units he wishes.

PREVIOUS PAGE IS BLANK

Input Format Requirements

Instead of using the usual fixed-format or free-form input capabilities of Simscript II, the input routine (named INITIALIZE) expects each input value to be preceded by a single "equals" sign (=) and followed by optional blank spaces. When searching the input data for a particular input value, the model ignores any text and numbers before the next equals sign. This allows the user to insert textual comments (excluding equals signs, of course) throughout the input, even if two or more input values appear on an input line.[1] In our use of the model, we simply identified input values by placing the Simscript variable name in front of each input, making it easier for us to change and identify inputs between simulation runs. Several examples of complete sets of inputs are shown below.

This way of formatting the model inputs should not be confused with some computer languages' free-form data input techniques where the program variables are identified, followed by values, in any order. Dyna-Sim still requires that a specific order within each input set be followed, as discussed below. In addition, the input routine performs no input checking.

System and Simulation Defining Parameters (Input Set 1)

Below are definitions of all of the system and simulation defining variables:

[1]See input examples later in the appendix.

Input Variable	Description
NTRIALS	Number of trials during entire simulation run
SVCDIST	Distribution of service times (1 = exponential, 2 = constant)
RULE	Service priority selection rule (1 = first come, first-serve, 2 = class with most jobs, 3 = differing priorities between job classes, 4 = job class with maximum number of backorders)
TINIT	Length of initial run-in
TRUNIN	Length of standard run-ins for each sample units
SAMPINT	Length of sample interval; controls sampling of units number in system statistics units
VARTOM	Variance-to-mean ratio of arrival process (≤ 1.0 for ordinary Poisson arrivals > 1.0 for compound Poisson arrivals)
NCLASSES	Number of job classes
NSERVERS	Number of servers
NPERIODS	Number of periods with differing job arrival rates plus one[a]

[a]NPERIODS is one greater than the number of periods with differing arrival rates to allow for the final run-in period described in Sec. II.

This input set is required for all four service rules. An example of these inputs for a five job class, two server simulation using service priority rule 1 is displayed in Fig. A.1. The number of periods specified (NPERIODS) includes the run-in interval. In the example, NPERIODS = 4 means that the trial is divided into three periods with an extra period for run-in.

Service Time Parameters (Input Set 2)

There are two available service time distributions currently built into Dyna-Sim, exponential and constant. The choice, as specified by the SVCDIST variable of input set 1, applies to all job classes. The user must specify one mean service time (in whatever time units are applicable) for each job class in this input set; the input variable (an array) is MEANST. If the service distribution is exponential, these input values represent the mean service time for each job class. If service times are constant, these inputs represent the constant service times. Either way, the means or constants can vary by job class.

Times for Changing Arrival Rates in Trials (Input Set 3)

As discussed in Section II, each trial in a simulation run can have time-varying job arrival rates. The NPERIODS variable of input set 1 defines the number of periods within a trial in which arrival rates may vary, plus one. Here, the user needs to indicate the times relative to the beginning of a trial that these arrival rates can change. The first input time should be zero (the first period starts at the beginning of a trial) and is assigned to variable A (an array). Succeeding inputs mark the times at which arrival rates can change. The last of these inputs should equal the length of a trial.

Arrival Rates for Each Job Class and Period (Input Set 4)

The model requires a job arrival rate for each each combination of job class and period. Because all job arrival times are determined with the negative exponential distribution, all of these rates must be greater than zero. The last arrival rate for each job class (M(1,4), M(2,4), etc. in Fig. A.1) is the arrival rate used for run-in intervals. Use a very small, positive fraction for the arrival rate if arrivals are to be precluded during a certain period; generally do this to the last period arrival rate in each job class because this rate will be used for the next run-in period. The program expects to find the rates in the order shown in Fig. A.1--by period within class.

Random Number Seeds (Input Set 5)

As described in Section II, the model deliberately uses different random number streams for arrivals and servicing within a job class. Consequently, for each job class, two random number seeds (integer valued) should be specified for input variable SEED.V (a Simscript array). The program expects to find two random number seeds even when constant service times are specified.

Priority Level of Each Job Class (Input Set 6)

This input set should be provided only if rule 3 (nonpreemptive job class priority levels) is specified in input set 1. The model allows more job classes than priority levels (when some job classes are at the same priority level), but the number of priority levels should never exceed the number of job classes.

As is the usual convention, priorities are related to class indices, with the lower number of the index having a higher priority level. The first input required to implement rule 3 is the number of priority levels (input variable NPRLEVELS). Then for each priority level, provide the highest index number of the job classes sharing that priority level (input variable FINPRLV, an array).

As an example, Fig. A.2 expands on our earlier example inputs by adding input set 6 where we desire to have job classes 1 and 2 at the highest priority, classes 3 and 4 at a lower priority, and finally class 5 at the lowest priority level.

Inventory Stock Levels for Each Job Class (Input Set 7)

This input set is needed only if rule 4 (priorities to jobs from job classes with the maximum backorder quantity) is specified in input set 1. The only input values needed are the initial inventory stock levels of items corresponding to each job class. These inputs are directed to input variable STOCK.LEVEL (an array), with one value for each job class. Figure A.3 shows how our earlier model input example might be modified to accommodate rule 4.

DYNA-SIM MODEL OUTPUTS

The first output report from Dyna-Sim is a replay of all model inputs. Using the input data from Fig. A.1, the model produces the output report shown in Fig. A.4. Although it doesn't contain much more information than the original input sets, it should be checked for each run to ensure that the model has read all input data properly. With the free-form input format and varying amount of data depending on system and simulation parameters, mistakes are possible. (For example, don't change the number of job classes from 4 to 3 and try to run without

dropping a mean service time, a pair of random number seeds, a job class from a priority level, and a stock level.) This report is generated as the inputs are obtained by the INITIALIZE routine.

If the variance-to-mean ratio (input set 1, VARTOM) of the arrival process is greater than 1.0, adjustments to the input arrival rates by job class and period are made as discussed in Sec. II. Had VARTOM been 10.0 instead of 1.0, the following display of the adjusted arrival rates would have been added to Fig. A.4:

ADJUSTED ARRIVAL RATES FOR UNDERLYING POISSON PROCESS PROCESS BY JOB CLASS AND PERIOD

	Period			
Class	1	2	3	4
1	.00	.78	.69	.00
2	.00	.54	.48	.00
3	.00	.45	.39	.00
4	.00	.30	.27	.00
5	.00	.05	.04	.00

The first summary output after all trials in the simulation have been completed is shown in Fig. A.5. This summarizes the relative frequency distributions of numbers of jobs in the system at the beginning and at each succeeding sample point across all trials. There is one set of distributions for each job class, followed by one for all classes combined. This and succeeding summary reports are generated by the ENDSIM routine.

Figure A.5 shows this summary report only for job class 3 using the inputs described in Fig. A.1 (service rule 1). For each sample point in a job class, the distributions are arranged vertically; labels giving the number of the sample point and the time within trials are shown below, along with the averages, maxima, and standard deviations. The distributions are truncated at 100 jobs, necessitating the reporting of the maximum. (The distribution in Fig. A.5 goes up to 20 because there were never more than 20 jobs in the system at any sample point.) This report provides sufficient information to easily establish the mode and median at a sample point using the frequency distribution. For example, at sample point 4, which shows an average of 6.54 jobs, the median is about 5.64 jobs and the mode is 5 jobs.

The next set of summaries report flow time statistics (time in system from job arrival to job completion. To do this, jobs are grouped by class and by the interval within a trial during which they arrive. There are also entries for jobs that arrived during run-in periods; these correspond to the last period shown in the display (period 4). The time in system statistics displayed for each period include the average, standard deviation, and number of jobs observed. Only 60 jobs arrived during run-in periods and no jobs arrived during first intervals of trials.

There are two separate displays of the type described above. The first displays flow time statistics for individual jobs that are measures of system performance. Figigure A.6 exemplifies this report for the rule 1 simulation discussed above. The second, flow time statistics for group means, is computed by taking the average flow time for each job class within each trial as the unit of observation instead of individual jobs. These averages can be assumed to be independent, so the standard deviations of the averages can be taken as the standard error of the flow time estimates. (An estimate of the standard error based on individual job flow times would give an underestimate because of the correlation in flow times for jobs that are in the system at the same time.) Figure A.7 is an example of this report for the same simulation run.

Unless rule 4 was specified, the output reporting ends here. For rule 4, which gives priority to jobs of the class currently experiencing the highest level of backorders, additional outputs related to backorders are provided. Figure A.8 shows the distribution, average, and standard deviation of the maximum over all classes of backorders at each of the sampling points. Figure A.9 shows, at each sampling point, the frequencies with which each class had the most backorders. For example, at sample point 3, job class 1 never had the maximum number of backorders; class 2 had the maximum 72 percent of the time. For purposes of invoking the maximum backorder rule, backorders were allowed to be negative. In reporting results, however, cases where stock is greater or equal to the number of jobs in the system are regarded as zero backorders.

```
  *** INPUT SET 1 -- SYSTEM AND SIMULATION DEFINING PARAMETERS ***


NCLASSES = 5
NSERVERS = 2
NPERIODS = 4
NTRIALS = 100
SVCDIST = 1
RULE = 1
TINIT = 0.0
TRUNIN = 1000.0
SAMPINT = 5.0
VARTOM = 1.0

  *** INPUT SET 2 -- SERVICE TIME PARAMETERS ***

MEANST(1) = 0.187   MEANST(2) = 0.471   MEANST(3) = 0.601
MEANST(4) = 0.384   MEANST(5) = 0.240


  *** INPUT SET 3 -- TIMES FOR CHANGING ARRIVAL RATES IN TRIAL ***

A(1)=0.0  A(2)=5.0  A(3)=12.0  A(4)=20.0


  *** INPUT SET 4 -- ARRIVAL RATES FOR EACH JOB CLASS, PERIOD ***

M(1,1)=0.0001  M(1,2)=3.06  M(1,3)=2.69  M(1,4)=0.0001
M(2,1)=0.0001  M(2,2)=2.11  M(2,3)=1.86  M(2,4)=0.0001
M(3,1)=0.0001  M(3,2)=1.74  M(3,3)=1.53  M(3,4)=0.0001
M(4,1)=0.0001  M(4,2)=1.18  M(4,3)=1.04  M(4,4)=0.0001
M(5,1)=0.0001  M(5,2)=.184  M(5,3)=.162  M(5,4)=0.0001

  *** INPUT SET 5 -- RANDOM NUMBER SEEDS ***

SEED.V(1)=4307  SEED.V(2)=9327
SEED.V(3)=3923  SEED.V(4)=3894
SEED.V(5)=8484  SEED.V(6)=1847
SEED.V(7)=3848  SEED.V(8)=8443
SEED.V(9)=2065  SEED.V(10)=7674
```

Fig. A.1--Example of inputs for rule 1 servicing priority

```
  *** INPUT SET 1 -- SYSTEM AND SIMULATION DEFINING PARAMETERS ***

NCLASSES = 5
NSERVERS = 2
NPERIODS = 4
NTRIALS = 100
SVCDIST = 1
RULE = 3
TINIT = 0.0
TRUNIN = 1000.0
SAMPINT = 5.0
VARTOM = 1.0

  *** INPUT SET 2 -- SERVICE TIME PARAMETERS ***

MEANST(1) = 0.187   MEANST(2) = 0.471   MEANST(3) = 0.601
MEANST(4) = 0.384   MEANST(5) = 0.240


  *** INPUT SET 3 -- TIMES FOR CHANGING ARRIVAL RATES IN TRIAL ***

A(1)=0.0  A(2)=5.0  A(3)=12.0  A(4)=20.0


  *** INPUT SET 4 -- ARRIVAL RATES FOR EACH JOB CLASS, PERIOD ***

M(1,1)=0.0001  M(1,2)=3.06  M(1,3)=2.69  M(1,4)=0.0001
M(2,1)=0.0001  M(2,2)=2.11  M(2,3)=1.86  M(2,4)=0.0001
M(3,1)=0.0001  M(3,2)=1.74  M(3,3)=1.53  M(3,4)=0.0001
M(4,1)=0.0001  M(4,2)=1.18  M(4,3)=1.04  M(4,4)=0.0001
M(5,1)=0.0001  M(5,2)=.184  M(5,3)=.162  M(5,4)=0.0001

  *** INPUT SET 5 -- RANDOM NUMBER SEEDS ***

SEED.V(1)=4307  SEED.V(2)=9327
SEED.V(3)=3923  SEED.V(4)=3894
SEED.V(5)=8484  SEED.V(6)=1847
SEED.V(7)=3848  SEED.V(8)=8443
SEED.V(9)=2065  SEED.V(10)=7674

  *** INPUT SET 6 -- PRIORITY LEVEL OF EACH JOB CLASS ***

NPRLEVELS = 3
FINPRLV(1) = 2   FINPRLV(2) = 4   FINPRLV(3)=5
```

Fig. A.2--Example of inputs for rule 3 servicing priority

```
  *** INPUT SET 1 -- SYSTEM AND SIMULATION DEFINING PARAMETERS ***

NCLASSES = 5
NSERVERS = 2
NPERIODS = 4
NTRIALS = 100
SVCDIST = 1
RULE = 4
TINIT = 0.0
TRUNIN = 1000.0
SAMPINT = 5.0
VARTOM = 1.0

  *** INPUT SET 2 -- SERVICE TIME PARAMETERS ***

MEANST(1) = 0.187   MEANST(2) = 0.471   MEANST(3) = 0.601
MEANST(4) = 0.384   MEANST(5) = 0.240


  *** INPUT SET 3 -- TIMES FOR CHANGING ARRIVAL RATES IN TRIAL ***

A(1)=0.0  A(2)=5.0  A(3)=12.0  A(4)=20.0


  *** INPUT SET 4 -- ARRIVAL RATES FOR EACH JOB CLASS, PERIOD ***

M(1,1)=0.0001  M(1,2)=3.06  M(1,3)=2.69  M(1,4)=0.0001
M(2,1)=0.0001  M(2,2)=2.11  M(2,3)=1.86  M(2,4)=0.0001
M(3,1)=0.0001  M(3,2)=1.74  M(3,3)=1.53  M(3,4)=0.0001
M(4,1)=0.0001  M(4,2)=1.18  M(4,3)=1.04  M(4,4)=0.0001
M(5,1)=0.0001  M(5,2)=.184  M(5,3)=.162  M(5,4)=0.0001

  *** INPUT SET 5 -- RANDOM NUMBER SEEDS ***

SEED.V(1)=4307  SEED.V(2)=9327
SEED.V(3)=3923  SEED.V(4)=3894
SEED.V(5)=8484  SEED.V(6)=1847
SEED.V(7)=3848  SEED.V(8)=8443


  *** INPUT SET 7 -- INVENTORY STOCK LEVELS FOR EACH JOB CLASS ***

STOCK.LEVEL(1)=2
STOCK.LEVEL(2)=4
STOCK.LEVEL(3)=4
STOCK.LEVEL(4)=4
STOCK.LEVEL(5)=1
```

Fig. A.3--Example of inputs for rule 4 servicing priority

DYNASIM INPUT DATA

SYSTEM AND SIMULATION DEFINING PARAMETERS --

NUMBER OF CLASSES	5	
NUMBER OF SERVERS	2	
NUMBER OF PERIODS	4	
NUMBER OF TRIALS	100	
DISTR OF SERVICE TIMES	1	
SERVICE RULE	1	1=FCFS, 2=CLASS WITH MOST IN QUEUE 3=CLASS PRIORITIES 4=CLASS WITH MAX BACKORDERS
LENGTH INITIAL RUNIN	0.	
LENGTH STANDARD RUNIN	1000.00	
LENGTH SAMPLE INTERVAL	5.00	
VARIANCE-TO-MEAN RATIO	1.00	

SERVICE PROCESSING TIME PARAMETERS --

JOB CLASS:	1	2	3	4	5
PARAMETER:	.187	.471	.601	.384	.240

TIMES FOR CHANGING ARRIVAL RATES WITHIN A TRIAL --

PERIOD:	1	2	3	4
TIME:	0.	5.00	12.00	20.00

ARRIVAL RATES BY JOB CLASS AND PERIOD --

CLASS	PERIOD			
	1	2	3	4
1	.00	3.06	2.69	.00
2	.00	2.11	1.86	.00
3	.00	1.74	1.53	.00
4	.00	1.18	1.04	.00
5	.00	.18	.16	.00

RANDOM NUMBER SEEDS --

CLASS	ARRIVAL	PROCESSING
1	4307	9327
2	3923	3894
3	8484	1847
4	3848	8443
5	2065	7674

Fig. A.4--Dyna-Sim output: Replay of inputs for rule 1 servicing priority example

RELATIVE FREQUENCIES FOR NUMBER OF CLASS 3 JOBS IN SYSTEM
AT EACH SAMPLE POINT COMPUTED ACROSS ALL TRIALS

NO. IN SYSTEM	SAMPLE POINT 1	2	3	4	5
20	0.	0.	0.	0.	.0100
19	0.	0.	0.	0.	0.
18	0.	0.	0.	.0100	.0300
17	0.	0.	0.	0.	0.
16	0.	0.	0.	.0100	.0300
15	0.	0.	0.	0.	.0400
14	0.	0.	0.	0.	0.
13	0.	0.	0.	.0200	.0400
12	0.	0.	0.	.0300	.0500
11	0.	0.	.0100	.0400	.0500
10	0.	0.	.0200	.0800	.1000
9	0.	0.	.0500	.0700	.1000
8	0.	0.	.0500	.1000	.0700
7	0.	0.	.0800	.1000	.1600
6	0.	0.	.0500	.1100	.1400
5	0.	0.	.1100	.1800	.0500
4	0.	0.	.1700	.0800	.0500
3	0.	0.	.1600	.0500	.0500
2	0.	0.	.1000	.0700	0.
1	0.	0.	.1100	.0400	.0300
0	1.0000	1.0000	.0900	.0100	0.

SAMPLE POINT	1	2	3	4	5
TIME IN TRIAL	0.	5.00	10.00	15.00	20.00
AVERAGE	0.	0.	4.04	6.54	8.54
MAXIMUM	0	0	11	18	20
STD DEV	0.	0.	2.70	3.32	3.93

Fig. A.5--Dyna-Sim summary output: Relative frequency distributions for number of jobs in system

FLOW TIME STATISTICS FOR INDIVIDUAL JOBS

	JOB CLASS 1			
PERIOD	1	2	3	4
AVERAGE	0.	2.359	5.956	.081
STD DEV	0.	1.929	2.720	.052
NUMBER	0	2153	2134	11

	JOB CLASS 2			
PERIOD	1	2	3	4
AVERAGE	.188	2.735	6.125	.706
STD DEV	0.	2.041	2.657	.533
NUMBER	1	1443	1451	14

	JOB CLASS 3			
PERIOD	1	2	3	4
AVERAGE	0.	2.801	6.220	.620
STD DEV	0.	2.020	2.725	.428
NUMBER	0	1247	1253	9

	JOB CLASS 4			
PERIOD	1	2	3	4
AVERAGE	0.	2.665	6.045	.452
STD DEV	0.	1.936	2.555	.493
NUMBER	0	851	806	9

	JOB CLASS 5			
PERIOD	1	2	3	4
AVERAGE	0.	2.743	5.817	.293
STD DEV	0.	2.225	2.564	.396
NUMBER	0	139	124	17

Fig. A.6--Dyna-Sim summary output: Flow time statistics for individual jobs

FLOW TIME STATISTICS FOR GROUP MEANS

	JOB CLASS 1			
PERIOD	1	2	3	4
AVERAGE	0.	2.311	5.815	.085
STD ERROR*		.118	.238	.015
NUMBER	0	100	100	9

	JOB CLASS 2			
PERIOD	1	2	3	4
AVERAGE	.188	2.561	6.062	.622
STD ERROR*	0.	.125	.234	.127
NUMBER	1	100	100	13

	JOB CLASS 3			
PERIOD	1	2	3	4
AVERAGE	0.	2.664	6.145	.553
STD ERROR*		.129	.236	.144
NUMBER	0	100	100	8

	JOB CLASS 4			
PERIOD	1	2	3	4
AVERAGE	0.	2.555	5.891	.308
STD ERROR*		.122	.234	.104
NUMBER	0	100	100	8

	JOB CLASS 5			
PERIOD	1	2	3	4
AVERAGE	0.	2.646	5.860	.203
STD ERROR*		.212	.279	.042
NUMBER	0	80	78	16

* STD ERROR = STD DEV / SQRT(NUMBER)
USE AS ESTIMATE OF STD DEV OF FLOW TIME SAMPLE MEANS

Fig. A.7--Dyna-Sim summary output: Flow time statistics for group means

RELATIVE FREQUENCIES FOR MAXIMUM BACKORDER QUANTITY ACROSS ALL JOB CLASSES IN SYSTEM AT EACH SAMPLE POINT COMPUTED OVER ALL TRIALS

MAXIMUM BOQ	SAMPLE POINT 1	2	3	4	5
20	0.	0.	0.	0.	.0100
19	0.	0.	0.	0.	0.
18	0.	0.	0.	0.	0.
17	0.	0.	0.	0.	0.
16	0.	0.	0.	0.	0.
15	0.	0.	0.	0.	.0200
14	0.	0.	0.	0.	.0100
13	0.	0.	0.	.0100	0.
12	0.	0.	0.	0.	.0100
11	0.	0.	0.	0.	.0200
10	0.	0.	0.	.0200	.0700
9	0.	0.	0.	.0200	.1000
8	0.	0.	0.	.0700	.0700
7	0.	0.	.0100	.0500	.0700
6	0.	0.	.0300	.0700	.1400
5	0.	0.	.0400	.0900	.1100
4	0.	0.	.0600	.1400	.1400
3	0.	0.	.0800	.1800	.0700
2	0.	0.	.1900	.0900	.0900
1	0.	0.	.1400	.1100	.0400
0	1.0000	1.0000	.4500	.1500	.0300

SAMPLE POINT	1	2	3	4	5
TIME IN TRIAL	0.	5.00	10.00	15.00	20.00
AVERAGE	-1.00	-1.00	1.26	3.64	5.99
MAXIMUM	-1	-1	7	13	20
STD DEV	0.	0.	1.94	2.85	3.54

Fig. A.8--Dyna-Sim summary output: Rule 4 maximum backorder quantity relative frequency distributions

RELATIVE FREQUENCIES FOR JOB CLASS WITH MAXIMUM BACKORDER QUANTITY COMPUTED OVER ALL TRIALS

MAXIMUM BOQ CLASS	SAMPLE POINT 1	2	3	4	5
5	0.	0.	.0100	.0100	.0100
4	0.	0.	.0500	.0500	.0200
3	0.	0.	.2200	.1400	.1500
2	0.	0.	.7200	.8000	.8200
1	0.	0.	0.	0.	0.

SAMPLE POINT	1	2	3	4	5
TIME IN TRIAL	0.	5.00	10.00	15.00	20.00

Fig. A.9--Dyna-Sim summary output: Rule 4 maximum backorder quantity job class relative frequency distributions

Appendix B

DYNA-SIM SIMSCRIPT II COMPUTER PROGRAM

This appendix presents some additional information about Dyna-Sim model, including the topics of programming language, variables, files, subprograms, event control, and guidelines for service selection rule addition. Then the complete Simscript source language program for the model is listed.

PROGRAMMING LANGUAGE

The entire Dyna-Sim model is written in the Simscript II language.[1] The specific implementation of the Simscript II language is named Simscript II.5 and is available from CACI, Inc.[2] The program source code listed at the end of the appendix was developed under the IBM MVS operating system, and any job control language statements shown there apply to that system.

VARIABLES

All system variables have explanatory comments appended to their definition in the PREAMBLE. Unless defined otherwise in the preamble or within a routine, all variables take the integer mode. Local variables of particular importance have explanatory comments in the routine where used.

SIMSCRIPT II SUBPROGRAMS

All of the various subprograms used in the Dyna-Sim model are listed below with a one or two line statement of their purpose. Subprograms are of three types in Simscript II--routines, functions, and events--a distinction preserved in this list.

---

[1] P.J. Kiviat et al., *SIMSCRIPT II Programming Language*, Prentice Hall, Englewood Cliffs, N.J., 1968.

[2] CACI, *SIMSCRIPT II.5 Reference Handbook*, Consolidated Analysis Centers Inc., Los Angeles, 1976.

Routines

MAIN	Entry point to the model; schedules first arrival in each job class and starts the simulation
INITIALIZE	Obtains all inputs and prepares input summary report
STARTSERVICE	Schedules an end of service EPROC event for a job
SELECT	Decides which job in the queue is to be served next, depending on service rule
TOFARR	Determines next time of arrival for a particular job class
ENDSIM	After the simulation terminates, this routine prepares statistical summary reports
TRACE	Used for debugging; tracks event timing
DUMPQUE	Used for debugging; prints contents of queue
GETINT	"Free-form" integer numeric input routine
GETREAL	"Free-form" real numeric input routine
FIND.EQUAL	Finds "=" sign for "free-form" input routines
LOAD.BUFFER	Obtains next input line for "free-form" inputs

Functions

PTSAMP	Samples processing times for jobs whenever a job arrives (never when a job goes into service)
SETPRIORITY	Sets priority of jobs for various rules

Events

ARRIVAL	Disposes of arriving jobs, placing them either in queue or service, and schedules next arrival
EPROC	Disposes of completed jobs and their servers; does some data collection
RUNIN	Schedules a STARTDATA event and the next RUNIN event
STARTDATA	Performs some initial data collection for a trial, then schedules the first sample of a trial
SAMPLE	Examines number in queue for each job class and backorder quantity for each job class for data collection, then schedules next sample in a trial

EVENT CONTROL

The main routine (MAIN) schedules the first RUN-IN event to happen after the initial run-in period. RUNIN schedules itself to be executed again after the next trial is over. It also schedules a STARTDATA event to happen when the run-in is complete. STARTDATA, which happens at the beginning of every trial, counts trials and is responsible for calling ENDSIM and stopping the simulation. STARTDATA also causes a SAMPLE event to occur immediately to sample number in system and backorder quantity statistics. SAMPLE then reschedules itself at the next sample point in the trial, if there is one.

GUIDELINES FOR ADDING NEW JOB PRIORITY RULES

If the user desires to add his own job priority rules, the following guidelines are presented. Waiting jobs are kept on a list (SIMSCRIPT set) ranked by an attribute of jobs called PRIORITY. All action is confined to two routines: SETPRIORITY and SELECT. When a job arrives, SETPRIORITY (a function) is invoked. Its purpose is to calculate the job's PRIORITY, which establishes how jobs in the queue are ordered. For rules 1 and 2, PRIORITY is the arrival time of the job. For rule 3, a table lookup based on the job class finds the corresponding priority level. The actual priority number is 100,000 times the priority level, plus the sequence number of the job divided by 100. This is to insure that FCFS will be the tie breaker when there are ties.

Routine SELECT performs the rest of the action. Whenever a server becomes free and there are jobs in the queue, SELECT is called by EPROC to find out which job should be worked on next. For rules 1 and 3, all SELECT has to do is report back the first job in the set. For rule 2, there is a global array called NUMQUE that tells how many jobs of each class are in the queue. The class of job to be selected is found by searching NUMQUE to find the element with the largest value, and then searching the queue to find the first job with the corresponding class. Rule 4 is handled similarly to rule 2, except that maximum backorder quantity is used in place of number in the queue, as described in Sec. II.

```
// JOB ...
// EXEC SIM25,PARM.SIM='M370,SEQ',REGION.GO=350K
//SIM.SYSIN DD *

PREAMBLE
NORMALLY MODE IS INTEGER
TEMPORARY ENTITIES
  EVERY JOB HAS
     AN ARRTIME,     ''ARRIVAL TIME
     A  TYPE,        ''CLASS OF JOB
     A  PROCTIME,    ''PROCESSING TIME
     A  PRIORITY,    ''TO BE USED IN IMPLEMENTING VARIOUS RULES
     AN ARRPRD       ''ARRIVAL PERIOD AS DEFINED BY THE A ARRAY
                     ''RELATED TO CHANGES IN ARRIVAL RATE,
                     ''USED TO GROUP JOBS FOR FLOW TIME STATISTICS
     AND BELONGS TO THE QUEUE

  DEFINE ARRTIME, PROCTIME, PRIORITY AS DOUBLE VARIABLES

  THE SYSTEM OWNS A QUEUE
  DEFINE QUEUE AS A SET RANKED BY LOW PRIORITY
     WITHOUT FL, FB, FA ROUTINES

PERMANENT ENTITIES
  EVERY SERVER HAS A BUSYIND

  ''THE FOLLOWING COMPOUND ENTITIES ARE FOR DATA COLLECTION BECAUSE
  ''   THE TALLY STATEMENT ONLY WORKS ON ATTRIBUTES, NOT VARIABLES

  EVERY CLASSDUM AND PERIODDUM HAS A FLOW AND AN AFLOW
  EVERY CLASSDUM AND SAMPDUM   HAS A NINSDC
  DEFINE FLOW AND AFLOW AS DOUBLE VARIABLES

  EVERY SAMPDUM HAS
    A MAXBQ,    '' MAX BACK ORDER QUANTITY
    A TRMAXBQ,  '' MAX OF MAXBQ AND 0 FOR DATA COLLECTION
    A CLMXBQ    '' MAX BOQ JOB CLASS

EVENT NOTICES
  INCLUDE
     RUNIN,     ''TO RUN SYSTEM BETWEEN TRIALS & AFTER INIT RUNIN
     STARTDATA ''BEGINNING OF A TRIAL
  EVERY
     SAMPLE HAS  ''TO TAKE DATA ON NUMBER IN SYSTEM
       A POINT   ''TO GROUP DATA BY SAMPLE POINT
  EVERY
     ARRIVAL HAS
       A  CLASS,   ''CLASS OF JOB ARRIVING
       A  PERIOD   ''TO MARK JOBS WITH GROUPING FOR FLOW TIME STATS
                   ''SEE ARRPRD IN JOB ENTITY
```

```
  EVERY
     EPROC HAS    ''END OF PROCESSING
       A ESERVER, ''SERVER INVOLVED
       A EJOB     ''ID OF JOB INVOLVED

PRIORITY ORDER IS RUNIN, STARTDATA, SAMPLE, ARRIVAL, EPROC

  DEFINE IDLE TO MEAN 0
  DEFINE BUSY TO MEAN 1
  DEFINE EXP  TO MEAN 1
  DEFINE CONST TO MEAN 2

  DEFINE SETPRIORITY, PTSAMP AS DOUBLE FUNCTIONS
  DEFINE NUMPARTS AS AN INTEGER FUNCTION

  DEFINE
    TINIT,     ''LENGTH OF INITIAL RUNIN
    TRUNIN,    ''LENGTH OF RUNIN AFTER INITIAL AND BETWEEN TRIALS
    TSTRUNIN, ''TIME STARTED LAST RUNIN
    TSTTRIAL, ''TIME LAST STARTED TRIAL WAS STARTED
    SAMPINT,  ''LENGTH OF A SAMPLING INTERVAL FOR NUMBER IN SYS STATS
    VARTOM,   ''VARIANCE TO MEAN RATIO (SHERBROOKE'S Q)
    LNQ,      ''LN(VARTOM)
    LOGPARM   ''PARAMETER (P/Q SHERBROOKE) FOR LOGARITHMIC DISTRIBUTION
        AS DOUBLE VARIABLES

  DEFINE MEANST AS A 1-DIM DOUBLE VARIABLE ''SVCE DISTN PARAMETERS

  DEFINE
    NPERIODS, ''NUMBER OF DIFFERENT ARRIVAL RATES, SEE A AND M
    NCLASSES, ''NUMBER OF JOB CLASSES
    NSERVERS, ''NUMBER OF SERVERS (REDUNDANT WITH N.SERVERS)
    NTRIALS,  ''NUMBER OF TRIALS TO MAKE
    SVCDIST,  ''DISTRIBUTION OF SERVICE TIMES, 1 = EXP, 2 = CONST
    RULE,     ''TO SELECT PRIORITY. 1 = FCFS, 2 = FIRST FROM CLASS
              ''   WITH MOST IN QUEUE. ADD OTHERS LATER
    SPNTS,    ''NUMBER OF SAMPLE POINTS IN A TRIAL
    STATUS,   ''0 = IN INITIAL RUNIN, 1 = IN A STANDARD RUNIN,
              ''2 = IN A TRIAL
    TRIALS,   ''NUMBER OF TRIALS SO FAR (COMPARE WITH NTRIALS)
    NARVLS,    ''NUMBER OF ARRIVALS SO FAR (FOR PRIORITY IN H-O-L)
    NPRLEVELS''NUMBER OF PRIORITY LEVELS FOR HEAD-OF-THE-LINE
        AS INTEGER VARIABLES

  DEFINE
    M         ''VECTORS OF ARRIVAL RATES, LAST FOR RUNIN
              '' SAME NUMBER OF RATES FOR ALL CLASSES
        AS A 2-DIM REAL VARIABLE

  DEFINE
    A         ''TIMES REL TO START OF A TRIAL FOR CHANGING ARRIVAL
              '' RATES. A(1) = 0.0,  A(NPERIODS) = LENGTH OF A TRIAL
        AS A 1-DIM DOUBLE VARIABLE
```

```
DEFINE
  NUMQUE,    ''NUMBER BY JOB CLASS OF JOBS IN QUEUE. FOR RULE 2
  NINS,      ''NUMBER IN SYSTEM BY JOB CLASS USED FOR DATA COLLECTION
  FINPRLV,   ''FIRST JOB CLASS IN PRIORITY LEVEL
  STOCK.LEVEL   '' STOCK LEVEL FOR EACH JOB CLASS (RULE 4)
       AS 1-DIM INTEGER VARIABLES

DEFINE
  BUFF.COLUMN   ''INPUT BUFFER COLUMN USED FOR INPUT SCANNING
   AS AN INTEGER VARIABLE

  ''DATA COLLECTION FOR FLOW TIMES

TALLY
  AVGFLOW AS THE GROUP AVERAGE,''WITHIN TRIAL FLOW TIME AVERAGES
  NAVGF AS THE GROUP NUMBER,
  MFLOW AS THE AVERAGE,
  VFLOW AS THE STD.DEV,
  NFLOW AS THE NUMBER
   OF FLOW

''THE IDEA IS TO TALLY FLOW WITHIN A TRIAL.  AT THE END OF A TRIAL,
''ASSIGN FLOW TO AFLOW AND RESET GROUP TOTALS OF FLOW. THEN WE GET
''AN ESTIMATE OF THE STANDARD DEVIATION OF THE SAMPLE AVERAGES BY
''TREATING THE AVERAGE WITHIN A TRIAL AS AN INDEPENDENT OBSERVATION.
''   FOR ESTIMATING FLOW TIMES, JOBS ARE GROUPED BY 'PERIODS'
''WHICH ARE TIMES OVER WHICH CONSTANT ARRIVAL RATES HOLD.
''SEE THE VARIABLES A, M, AND NPERIODS.

TALLY
  GRPFLOW AS THE MEAN,
  VGRPFLOW AS THE STD.DEV,
  NGRPFLOW AS THE NUMBER
    OF AFLOW

''DATA COLLECTION FOR NUMBER IN SYSTEM
'' NINSDC IS THE  NUMBER IN SYSTEM BY CLASS AND SAMPLE POINT

TALLY
  HISTINSYS(0 TO 101 BY 1) AS THE HISTOGRAM,
  MAXINSYS  AS THE MAXIMUM,
  AVGINSYS  AS THE AVERAGE,
  SDINSYS AS THE STD.DEV,
  NUMINSYS AS THE NUMBER
    OF NINSDC

''DATA COLLECTION FOR MAX BACKORDER QUANTITY
'' MAXBQ IS THE MAX BACKORDER QUANTITY BY SAMPLE POINT
```

```
  TALLY
    HISTMAXBQ(0 TO 101 BY 1) AS THE HISTOGRAM,
    MAXXBQ  AS THE MAXIMUM,
    AVGMAXBQ  AS THE AVERAGE,
    SDMAXBQ AS THE STD.DEV,
    NUMMAXBQ AS THE NUMBER
      OF TRMAXBQ

''DATA COLLECTION FOR MAX BACKORDER QUANTITY JOBCLASS
'' CLMXBQ IS THE MAX BACKORDER QUANTITY JOBCLASS BY SAMPLE POINT

  TALLY
    HISTCLMXBQ(0 TO 101 BY 1) AS THE HISTOGRAM
      OF CLMXBQ

END ''OF PREAMBLE

MAIN
    CALL INITIALIZE
    ''THE FOLLOWING ST. CAUSES TRACE TO BE CALLED FROM TIMING ROUTINE
    ''DEBUG    LET BETWEEN.V = 'TRACE'
    FOR I = 1 TO NCLASSES
         SCHEDULE AN ARRIVAL GIVEN I, NPERIODS NOW
    LET STATUS = 0
    SCHEDULE A RUNIN AT TINIT
    START SIMULATION
END '' OF MAIN

ROUTINE INITIALIZE
  DEFINE XIN AS A REAL VARIABLE
  LET BUFF.COLUMN=-1  ''INITIALIZE FOR FIRST RECORD
  USE UNIT 03 FOR INPUT
  WRITE AS B 20, "DYNASIM INPUT DATA",/ ,/

 '' ********** INPUT SET 1
  CALL GETINT YIELDING NCLASSES
  CALL GETINT YIELDING NSERVERS
  CALL GETINT YIELDING NPERIODS
  CALL GETINT YIELDING NTRIALS
  CALL GETINT YIELDING SVCDIST
  CALL GETINT YIELDING RULE
  CALL GETREAL YIELDING TINIT
  CALL GETREAL YIELDING TRUNIN
  CALL GETREAL YIELDING SAMPINT
  CALL GETREAL YIELDING VARTOM
```

```
PRINT 15 LINES WITH NCLASSES, NSERVERS, NPERIODS, NTRIALS, SVCDIST,
   RULE,  TINIT, TRUNIN, SAMPINT, VARTOM THUS
SYSTEM AND SIMULATION DEFINING PARAMETERS --

  NUMBER OF CLASSES        ***
  NUMBER OF SERVERS        ***
  NUMBER OF PERIODS        ***
  NUMBER OF TRIALS         ***
  DISTR OF SERVICE TIMES   ***
  SERVICE RULE             ***      1=FCFS, 2=CLASS WITH MOST IN QUEUE
                                    3=CLASS PRIORITIES
                                    4=CLASS WITH MAX BACKORDERS
  LENGTH INITIAL RUNIN     *****.**
  LENGTH STANDARD RUNIN    *****.**
  LENGTH SAMPLE INTERVAL   *****.**

  VARIANCE-TO-MEAN RATIO   *****.**

  RESERVE
    NUMQUE(*), MEANST(*) AS NCLASSES,
    A(*)       AS NPERIODS,
    M(*,*) AS NCLASSES BY NPERIODS

 '' ************** INPUT SET 2
  FOR I = 1 TO NCLASSES, DO
    CALL GETREAL YIELDING XIN
    LET MEANST(I)=XIN
  LOOP
  SKIP 2 LINES
  WRITE AS "SERVICE PROCESSING TIME PARAMETERS --"
  WRITE 1 AS /, "  JOB CLASS:", I 7
  FOR I = 2 TO NCLASSES WRITE I AS I 10
  WRITE AS /, "  PARAMETER:"
  FOR I = 1 TO NCLASSES WRITE MEANST(I) AS D(10,3)

 '' ************** INPUT SET 3
  FOR I = 1 TO NPERIODS, DO
    CALL GETREAL YIELDING XIN
    LET A(I)=XIN
  LOOP
  SKIP 3 OUTPUT LINES
  WRITE AS "TIMES FOR CHANGING ARRIVAL RATES WITHIN A TRIAL --"
  WRITE 1 AS /, "  PERIOD:", I 8
  FOR J = 2 TO NPERIODS WRITE J AS I 10
  WRITE AS /, "  TIME:  "
  FOR J = 1 TO NPERIODS WRITE A(J) AS D(10,2)
```

```
'' ********** INPUT SET 4
 SKIP 3 LINES
 WRITE AS "ARRIVAL RATES BY JOB CLASS AND PERIOD --",
   /, "  CLASS                PERIOD"
 WRITE 1 AS /, I 13
 FOR I = 2 TO NPERIODS WRITE I AS I 10
 WRITE AS /
 FOR J = 1 TO NCLASSES DO
   WRITE J AS I 5
   FOR K = 1 TO NPERIODS DO

    CALL GETREAL YIELDING XIN
    LET M(J,K)=XIN
    WRITE M(J,K) AS D(10,2)
   LOOP
   SKIP 1 LINE
 LOOP

 IF VARTOM GT 1.0, CALL ARRPARMS  '' TO ADJUST M
 ALWAYS

'' ********** INPUT SET 5
 ''READ RN SEEDS: for ARRIVAL AND PROCT FOR EACH CLASS
 RELEASE SEED.V
 RESERVE SEED.V(*) AS 2 * NCLASSES
 SKIP 1 LINE
 WRITE AS /, "RANDOM NUMBER SEEDS --",
          /,"  CLASS     ARRIVAL   PROCESSING", /
 FOR J = 1 TO NCLASSES DO
   CALL GETREAL YIELDING XIN  LET SEED.V(J)=XIN
   CALL GETREAL YIELDING XIN  LET SEED.V(J+NCLASSES)=XIN
   WRITE J, SEED.V(J), SEED.V(J+NCLASSES) AS I 5, I 12, I 12, /
 LOOP

'' ********** INPUT SET 6  (RULE 3 ONLY)
 '' IF HEAD OF LINE, READ NUMBER OF PRIORITY LEVELS AND FIRST
 ''   JOB CLASS FOR EACH PRIORITY CLASS
 IF RULE EQ 3
   CALL GETINT YIELDING NPRLEVELS
   RESERVE FINPRLV(*) AS NPRLEVELS
   FOR I = 1 TO NPRLEVELS, DO
     CALL GETREAL YIELDING XIN
     LET FINPRLV(I)=XIN
   LOOP
   WRITE AS /,/,"PRIORITY LEVELS --      "
   WRITE NPRLEVELS AS /, "NUMBER OF PRIORITY LEVELS =", I 4,/
   WRITE AS /, "  PRIORITY LEVEL INDEX:                "
   FOR I = 1 TO NPRLEVELS WRITE I AS I 6
   WRITE AS /, "  FIRST JOB CLASS IN PRIORITY LEVEL:"
   FOR I = 1 TO NPRLEVELS WRITE FINPRLV(I) AS I 6
   WRITE AS /
```

```
  ALWAYS
'' ********** INPUT SET 7  (RULE 4 ONLY)
  IF RULE EQ 4
    RESERVE STOCK.LEVEL(*) AS NCLASSES
    WRITE AS /,/, "STOCK LEVELS FOR EACH JOB CLASS --"
    WRITE AS /, "  CLASS    LEVEL", /
    FOR J=1 TO NCLASSES DO
      CALL GETREAL YIELDING XIN  LET STOCK.LEVEL(J)=XIN
      WRITE J, STOCK.LEVEL(J) AS /,I 5, I 10
    LOOP

  ALWAYS
  CREATE EACH SERVER(NSERVERS)
  RESERVE NINS AS NCLASSES + 1
  LET SPNTS = 1 + TRUNC.F(A(NPERIODS) / SAMPINT)
  CREATE EACH CLASSDUM(NCLASSES + 1)
  CREATE EACH PERIODDUM(NPERIODS)
  CREATE EACH SAMPDUM(SPNTS)
  LET TRIALS = 0
  START NEW PAGE
  RETURN
END ''OF INITIALIZATION



ROUTINE ARRPARMS
  DEFINE QM1 AND LFACTOR AS DOUBLE VARIABLES

  LET QM1 = VARTOM-1.0
  LET LNQ = LOG.E.F(VARTOM)
  LET LFACTOR=LNQ/QM1
  LET LOGPARM = QM1/VARTOM

  FOR I=1 TO NCLASSES
    FOR J=1 TO NPERIODS, DO
      LET M(I,J) = M(I,J)*LFACTOR
  LOOP

  SKIP 3 LINES
  WRITE AS "ADJUSTED ARRIVAL RATES FOR UNDERLYING",
      " POISSON PROCESS BY JOB CLASS AND PERIOD --",
    /, "  CLASS                PERIOD"
  WRITE 1 AS /, I 13
  FOR I = 2 TO NPERIODS WRITE I AS I 10
  WRITE AS /
  FOR J = 1 TO NCLASSES DO
    WRITE J AS I 5
    FOR K = 1 TO NPERIODS DO
     WRITE M(J,K) AS D(10,2)
    LOOP
    SKIP 1 LINE
  LOOP
END ''OF ARRPARMS
```

```
EVENT ARRIVAL(KIND, PERIOD) SAVING THE EVENT NOTICE

  DEFINE TOA AS A DOUBLE VARIABLE

  LET JOBSHERE = 1
  IF VARTOM GT 1.0, LET JOBSHERE = NUMPARTS(KIND)
  ALWAYS

  FOR PART=1 TO JOBSHERE, DO

    LET NARVLS = NARVLS + 1
    CREATE JOB
      LET ARRTIME(JOB) = TIME.V
      LET TYPE(JOB) = KIND
      LET PROCTIME(JOB) = PTSAMP(KIND)
      LET ARRPRD(JOB) = PERIOD
      LET PRIORITY(JOB) = SETPRIORITY(JOB)
      LET NINS(KIND) = NINS(KIND) + 1
      LET NINS(NCLASSES+1) = NINS(NCLASSES+1) + 1

    ''SEE IF THERE IS A FREE SERVER FOR THE NEW JOB

    LET GO = 0
    FOR EACH SERVER CALLED I, WITH BUSYIND(I) = IDLE,
      FIND THE FIRST CASE
      IF FOUND LET GO = 1
      ALWAYS
    IF GO = 1 CALL STARTSERVICE GIVEN I, JOB
    ALWAYS
    IF GO = 0
      FILE JOB IN QUEUE
      LET NUMQUE(KIND) = NUMQUE(KIND) + 1
    ALWAYS

  ''DEBUG     THE FOLLOWING IS FOR DEBUGGING
  ''DEBUG PRINT 1 LINE THUS
  ''DEBUG JOB    ARRTIME    TYPE   PROCTIME ARRPRD GO M.QUEUE NINS(TYPE)
  ''DEBUG PRINT 1 LINE WITH JOB, ARRTIME(JOB), TYPE(JOB), PROCTIME(JOB),
    ''DEBUG ARRPRD(JOB), GO, M.QUEUE(JOB), NINS(KIND) LIKE THIS
''DEBUG ********** *****.******** *   ****.********   *    *    *    ****

  LOOP

  ''SCHEDULE THE NEXT ARRIVAL

  CALL TOFARR GIVEN KIND YIELDING TOA, JPERIOD
''DEBUG   WRITE TIME.V, TOA AS "TIME.V=",D(14,8),S 4,"TOA=",D(14,8),/
  SCHEDULE THE ARRIVAL GIVEN KIND, JPERIOD AT TOA

  RETURN
END  ''OF ARRIVAL
```

```
FUNCTION NUMPARTS GIVEN JCLASS

  DEFINE BY, SUM, TEST AS DOUBLE VARIABLES

  LET COUNT = 1
  LET BY = LOGPARM
  LET SUM = BY
  LET TEST = LNQ * RANDOM.F(JCLASS)
  WHILE SUM LT TEST DO
    LET COUNT = COUNT + 1
    LET BY = BY * LOGPARM
    LET SUM = SUM + BY / COUNT
  LOOP
  RETURN WITH COUNT
END



EVENT EPROC GIVEN ISERVER, JOBID

  ''DO ACCOUNTING FOR DATA COLLECTION

  LET K = TYPE(JOBID)
  LET FLOW(K, ARRPRD(JOBID)) = TIME.V - ARRTIME(JOBID)
  LET NINS(K) = NINS(K) - 1
  LET NINS(NCLASSES+1) = NINS(NCLASSES+1) - 1

  ''DEBUG   WRITE JOBID, K, ARRPRD(JOBID), TIME.V - ARRTIME(JOBID)
  ''DEBUG    AS /, "EPROC  JOBID=", I 8, S 4,"CLASS=", I 4,
  ''DEBUG       S 4, "ARRPRD=",I 4, S 4, "FLOW=", D(14,8), /,/

  ''DISPOSE OF JOB, SERVER

  LET BUSYIND(ISERVER) = IDLE
  DESTROY JOB CALLED JOBID
  IF QUEUE IS NOT EMPTY
    CALL SELECT YIELDING JOBID
    REMOVE JOBID FROM QUEUE
    LET NUMQUE(TYPE(JOBID)) = NUMQUE(TYPE(JOBID)) - 1
    CALL STARTSERVICE GIVEN ISERVER, JOBID
  ALWAYS
  RETURN
END  ''OF EPROC


EVENT RUNIN SAVING THE EVENT NOTICE
  LET TSTRUNIN = TIME.V
  LET STATUS = 1
  SCHEDULE A STARTDATA IN TRUNIN DAYS
  SCHEDULE THIS RUNIN IN TRUNIN + A(NPERIODS) DAYS
  RETURN
END  ''OF RUNIN
```

```
EVENT STARTDATA
  LET STATUS = 2
  LET TSTTRIAL = TIME.V

  '' DATA COLLECTION FOR FLOW TIMES

  IF TRIALS GT 0
    FOR K = 1 TO NCLASSES
      FOR L = 1 TO NPERIODS DO
        IF NAVGF(K,L) GT 0
    ''DEBUG          WRITE TIME.V, K, L, AVGFLOW(K,L), NAVGF(K,L) AS
    ''DEBUG           "TIME.V=",D(10,4), S 4, "K,L=", I 3, I 3, S 4,
    ''DEBUG           "AVGFLOW=", D(10,4), S 4, "NAVGF=", I 4 ,/
          LET AFLOW(K,L) = AVGFLOW(K,L)
        ALWAYS
      LOOP
  ALWAYS
  FOR EACH CLASSDUM  FOR EACH PERIODDUM
        RESET GROUP TOTALS OF FLOW

  ''COUNT TRIALS AND TEST FOR END OF SIMULATION

  LET TRIALS = TRIALS + 1
  IF TRIALS GT NTRIALS
    CALL ENDSIM
    STOP

  ''NOT DONE, CONTINUE MORE TRIALS

  ELSE
  SCHEDULE A SAMPLE GIVEN 0 NOW
  RETURN
END  ''OF STARTDATA


EVENT SAMPLE GIVEN POINT SAVING THE EVENT NOTICE
  LET POINT = POINT + 1
  FOR K = 1 TO NCLASSES + 1 LET NINSDC(K,POINT) = NINS(K)

  LET TEMP.MAXBQ=-999999    LET TEMP.CLMXBQ=0
  FOR K=1 TO NCLASSES  DO
    LET DIFF=NINS(K)-STOCK.LEVEL(K)
    IF DIFF GT TEMP.MAXBQ
      LET TEMP.MAXBQ=DIFF  LET TEMP.CLMXBQ=K
    ALWAYS
  LOOP
  LET MAXBQ(POINT)=TEMP.MAXBQ
  LET TRMAXBQ(POINT) = MAX.F(0, TEMP.MAXBQ)
  LET CLMXBQ(POINT)=TEMP.CLMXBQ
```

```
  ''DEBUG  WRITE TIME.V,POINT,MAXBQ(POINT),CLMXBQ(POINT)
  ''DEBUG        AS /,"SAMPLE AT ",D(10,3),3 I 5

  ''DEBUG   WRITE POINT AS "NINSDC   POINT=", I 4, /
  ''DEBUG   FOR K = 1 TO NCLASSES WRITE NINSDC(K,POINT) AS I 8
  ''DEBUG   WRITE AS /,/

  IF POINT LT SPNTS
    SCHEDULE THIS SAMPLE GIVEN POINT IN SAMPINT DAYS
    RETURN
  ELSE DESTROY SAMPLE
  RETURN
END  '' OF SAMPLE


ROUTINE STARTSERVICE GIVEN ISERVER, JOBID
  LET BUSYIND(ISERVER) = BUSY
  SCHEDULE AN EPROC GIVEN ISERVER, JOBID IN PROCTIME(JOBID) DAYS
  RETURN
END  ''OF STARTSERVICE


FUNCTION PTSAMP GIVEN KIND  ''FOR SAMPLING SERVICE TIMES
  IF SVCDIST EQ EXP
    RETURN WITH EXPONENTIAL.F(MEANST(KIND), KIND + NCLASSES)
  ELSE
    RETURN WITH MEANST(KIND)
END  ''OF PTSAMP


FUNCTION SETPRIORITY(JOBID)  ''SETS PRIORITY OF JOBS FOR VARIOUS RULES
  IF RULE EQ 1 OR RULE EQ 2    ''FCFS OR FIRST JOB FROM CLASS
       OR RULE EQ 4

                               ''  WITH MAX NO. WAITING
    RETURN WITH ARRTIME(JOBID)
  ELSE

  IF RULE EQ 3
    LET JCLASS = TYPE(JOBID)
    LET I = 1
    WHILE JCLASS GT FINPRLV(I) UNTIL I = NPRLEVELS
      LET I = I + 1
    RETURN WITH REAL.F(100000 * I) + REAL.F(NARVLS)/100.0

  ELSE
    RETURN WITH 0.0
END  '' OF SETPRIORITY
```

```
ROUTINE SELECT YIELDING JOBID
 ''DEBUG  WRITE TIME.V AS /,"ENTER SELECT ROUTINE AT TIME.V=",D(10,3),/
 ''DEBUG   CALL DUMPQUE
 ''DEBUG   IF TIME.V GT 100.0 STOP
 ''DEBUG   ELSE
  IF RULE EQ 1    ''FCFS
    OR RULE EQ 3  ''HEAD OF THE LINE
    LET JOBID = F.QUEUE
    RETURN
  ELSE
  IF RULE = 2  ''EARLIEST ARRIVING JOB FROM CLASS WITH MOST WAITING
   FOR I = 1 TO NCLASSES COMPUTE LONGEST = MAXIMUM(I) OF NUMQUE(I)
   FOR EACH JOBID OF QUEUE WITH TYPE(JOBID) = LONGEST
     FIND THE FIRST CASE
    RETURN
  ELSE
  IF RULE = 4  ''EARLIEST ARRIVING JOB FROM CLASS WITH HIGHEST
               '' BACKORDER LEVEL (NO. WAITING JOBS - STOCK LEVEL)
 ''DEBUG   WRITE AS /,"SELECT RULE 4"
 ''DEBUG   WRITE AS /,"NUMQUE(*)"
 ''DEBUG   FOR KK=1 TO NCLASSES WRITE NUMQUE(KK) AS I 5
   FOR I = 1 TO NCLASSES DO
    IF NUMQUE(I) GT 0
     COMPUTE MAXBO = MAXIMUM(I) OF NUMQUE(I)-STOCK.LEVEL(I)
 ''DEBUG     WRITE I,NUMQUE(I)-STOCK.LEVEL(I),MAXBO AS /,"FIND MAXBO",3 I 5
    ALWAYS
   LOOP
 ''DEBUG    WRITE MAXBO AS /,"MAXBO=",I 5
   FOR EACH JOBID OF QUEUE WITH TYPE(JOBID) = MAXBO
     FIND THE FIRST CASE
  ALWAYS
 ''DEBUG    WRITE JOBID AS /,"JOBID=",I 15
  RETURN
END  ''OF SELECT


ROUTINE TOFARR ''TIME OF ARRIVAL'' GIVEN KIND YIELDING TOA, JPERIOD
'' FOR SAMPLING ARRIVAL TIMES

  DEFINE TOA, B, Z, T AS DOUBLE VARIABLES

  LET Z = EXPONENTIAL.F(1.0,KIND)

  ''DEBUG   WRITE TIME.V AS /,"TIME.V= ",D(14,8),/
  ''DEBUG     WRITE STATUS, Z AS "IN TOFARR WITH STATUS=", I 2, S 4,
  ''DEBUG        "Z=", D(14,8), /
```

```
IF STATUS EQ 0    ''IN THE INITIAL RUNIN
  LET TOA = TIME.V + Z / M(KIND, NPERIODS)
  LET JPERIOD = NPERIODS
  IF TOA LT TINIT
    RETURN
  ELSE
    LET Z = Z - (TINIT - TIME.V) * M(KIND, NPERIODS)
    LET T = TINIT
    LET B = T + TRUNIN

ALWAYS
IF STATUS EQ 1    ''IN A STANDARD RUNIN
  LET B = TSTRUNIN + TRUNIN
  LET JPERIOD = NPERIODS
  LET T = TIME.V

  ''DEBUG     WRITE JPERIOD, B AS "STATUS IS 1, JPERIOD=", I 3, S 4,
  ''DEBUG         "B=", D(14,8), /

ALWAYS
IF STATUS EQ 2  ''IN A TRIAL, HAVE TO LOCATE JPERIOD
  LET JPERIOD = 1
  LET T = TSTTRIAL
  WHILE TIME.V GT T + A(JPERIOD + 1) DO
    LET JPERIOD = JPERIOD + 1
    IF JPERIOD GT NPERIODS
      LET T = T + A(NPERIODS)
      LET JPERIOD = 1
    ALWAYS
   LOOP
   LET B = T + A(JPERIOD + 1)
  LET T = TIME.V

   ''DEBUG     WRITE JPERIOD, B, T AS "LEAVING STATUS=2 PART",
   ''DEBUG    "  JPERIOD=", I 3, S 4, "B=", D(14,8), "T=", D(14,8), /

ALWAYS

'' WE NOW HAVE B = TIME OF START OF THE FIRST PERIOD BEYOND
'' TIME.V AND JPERIOD = INDEX OF THE PERIOD CONTAINING TIME.V

''DEBUG     WRITE JPERIOD, B, T, Z AS "BEFORE UNTIL  JPERIOD=", I 3,
''DEBUG      S 4, "B=", D(14,8), S 4, "T=", D(14,8), S 4, "Z=",
''DEBUG      D(14,8), /
```

```
  UNTIL Z LT (B - T) * M(KIND, JPERIOD) DO
    LET Z = Z - (B - T) * M(KIND, JPERIOD)
    LET T = B
    LET JPERIOD = JPERIOD + 1
    IF JPERIOD LT NPERIODS
      LET B = B + A(JPERIOD + 1) - A(JPERIOD)
    ALWAYS
    IF JPERIOD EQ NPERIODS
      LET B = B + TRUNIN
    ALWAYS
    IF JPERIOD GT NPERIODS
      LET B = B + A(2)
      LET JPERIOD = 1
    ALWAYS
    ''DEBUG   WRITE Z AS "IN UNTIL  Z REDUCED TO", D(14,8), /
  LOOP

  LET TOA = T + Z / M(KIND, JPERIOD)

  ''DEBUG       WRITE T, Z, TOA, JPERIOD AS "LEAVING TOFARR WITH T=",
  ''DEBUG        D(14,8), S 4, "Z=", D(14,8), S 4, "TOA=", D(14,8),
  ''DEBUG        S 4, "JPERIOD=", I 3, /, /

  RETURN
END  ''OF TOFARR


ROUTINE ENDSIM
  DEFINE STDERR AS A REAL VARIABLE

  WRITE TIME.V AS *, "ENDRUN SIMULATION ENDED AT TIME", D(18,6), /, /
```

```
FOR K = 1 TO NCLASSES + 1 DO
  SKIP 5 OUTPUT LINES
  IF K LE NCLASSES
    WRITE K AS "RELATIVE FREQUENCIES FOR NUMBER OF CLASS",
      I 3, " JOBS IN SYSTEM", /,
      "    AT EACH SAMPLE POINT COMPUTED ACROSS ALL TRIALS", /,/
  ALWAYS
  IF K GT NCLASSES
    WRITE AS "RELATIVE FREQUENCIES FOR OVERALL NUMBER OF JOBS",
      " IN SYSTEM",/,
      "    AT EACH SAMPLE POINT COMPUTED ACROSS ALL TRIALS", /,/
  ALWAYS
  WRITE AS "    NO. IN                SAMPLE POINT", /,
           "    SYSTEM", B 14
  FOR L = 1 TO SPNTS WRITE L AS I 8
  WRITE AS /,/
  LET J = 101 LET N = 0
  WHILE N EQ 0, UNTIL J EQ 1, DO
    LET J = J - 1
    FOR L = 1 TO SPNTS LET N = N + HISTINSYS(K,L,J+1)
    ''DEBUG IF K EQ 4
    ''DEBUG   WRITE J, N AS "J=", I 5, "  N=", I 5, /  ALWAYS
  LOOP
  FOR JJ BACK FROM J TO 0 DO
    WRITE JJ AS I 9 WRITE AS B 18
    FOR L = 1 TO SPNTS
     WRITE REAL.F(HISTINSYS(K, L, JJ+1)) /
      REAL.F(NUMINSYS(K,L)) AS D(8,4)
    WRITE AS /
  LOOP

WRITE AS "---------"
FOR L = 1 TO SPNTS + 1 WRITE AS "--------"
WRITE AS /, /, "SAMPLE POINT", B 14
FOR L = 1 TO SPNTS WRITE L AS I 8  WRITE AS /, "TIME IN TRIAL"
WRITE 0.0 AS D(10,2)
FOR L = 1 TO SPNTS - 1 WRITE L * SAMPINT AS D(8,2)  WRITE AS /, /
WRITE AS "AVERAGE        "
FOR L = 1 TO SPNTS WRITE AVGINSYS(K, L) AS D(8,2)
WRITE AS /, "MAXIMUM     "
FOR L = 1 TO SPNTS WRITE MAXINSYS(K,L) AS I 8
WRITE AS /, "STD DEV         "
FOR L = 1 TO SPNTS WRITE SDINSYS(K,L) AS D(8,2)
WRITE AS /, /, /
LOOP  ''END OF NUMBER IN SYSTEM DATA
```

```
IF RULE=4
  START NEW PAGE
  WRITE AS "RELATIVE FREQUENCIES FOR MAXIMUM BACKORDER QUANTITY",
           " ACROSS ALL JOB", /,
           "  CLASSES IN SYSTEM AT EACH SAMPLE POINT COMPUTED OVER",
           " ALL TRIALS", /,/
  WRITE AS "   MAXIMUM                SAMPLE POINT", /,
           "     BOQ  ", B 14
  FOR L = 1 TO SPNTS WRITE L AS I 8
  WRITE AS /,/
  LET J = 101 LET N = 0
  WHILE N EQ 0, UNTIL J EQ 1, DO
    LET J = J - 1
    FOR L = 1 TO SPNTS LET N = N + HISTMAXBQ(L,J+1)
  LOOP
  FOR JJ BACK FROM J TO 0 DO
    WRITE JJ AS I 9 WRITE AS B 18
    FOR L = 1 TO SPNTS
     WRITE REAL.F(HISTMAXBQ(L,JJ+1)) /
      REAL.F(NUMMAXBQ(L)) AS D(8,4)
    WRITE AS /
  LOOP

  WRITE AS "---------"
  FOR L = 1 TO SPNTS + 1 WRITE AS "--------"
  WRITE AS /, /, "SAMPLE POINT", B 14
  FOR L = 1 TO SPNTS WRITE L AS I 8  WRITE AS /, "TIME IN TRIAL"
  WRITE 0.0 AS D(10,2)
  FOR L = 1 TO SPNTS - 1 WRITE L * SAMPINT AS D(8,2)  WRITE AS /, /
  WRITE AS "AVERAGE       "
   FOR L = 1 TO SPNTS WRITE AVGMAXBQ(L) AS D(8,2)
  WRITE AS /, "MAXIMUM     "
   FOR L = 1 TO SPNTS WRITE MAXXBQ(L) AS I 8
  WRITE AS /, "STD DEV       "
   FOR L = 1 TO SPNTS WRITE SDMAXBQ(L) AS D(8,2)
```

```
SKIP 5 OUTPUT LINES
WRITE AS "RELATIVE FREQUENCIES FOR JOB CLASS WITH MAXIMUM", /,
        "  BACKORDER QUANTITY COMPUTED OVER ALL TRIALS", /,/
WRITE AS "  MAXIMUM                    SAMPLE POINT", /,
         " BOQ CLASS", B 14
FOR L = 1 TO SPNTS WRITE L AS I 8
WRITE AS /,/
LET J = 101 LET N = 0
WHILE N EQ 0, UNTIL J EQ 1, DO
  LET J = J - 1
  FOR L = 1 TO SPNTS LET N = N + HISTMAXBQ(L,J+1)
LOOP
FOR JJ BACK FROM NCLASSES TO 1 DO
  WRITE JJ AS I 5 WRITE AS B 18
  FOR L = 1 TO SPNTS
   WRITE REAL.F(HISTCLMXBQ(L,JJ)) /
    REAL.F(NTRIALS) AS D(8,4)
  WRITE AS /
LOOP

WRITE AS "---------"
FOR L = 1 TO SPNTS + 1 WRITE AS "--------"
WRITE AS /, /, "SAMPLE POINT", B 14
FOR L = 1 TO SPNTS WRITE L AS I 8  WRITE AS /, "TIME IN TRIAL"
WRITE 0.0 AS D(10,2)
FOR L = 1 TO SPNTS - 1 WRITE L * SAMPINT AS D(8,2)  WRITE AS /, /

ALWAYS
START NEW PAGE
WRITE AS B 5, "FLOW TIME STATISTICS FOR INDIVIDUAL JOBS", /, /
FOR K = 1 TO NCLASSES DO
  WRITE K AS S 10, "   JOB CLASS", I 3, /
  WRITE AS "PERIOD  "
  FOR L = 1 TO NPERIODS WRITE L AS I 8
  WRITE AS /
  WRITE AS "AVERAGE   "
  FOR L = 1 TO NPERIODS WRITE MFLOW(K,L) AS D(8,3)
  WRITE AS /, "STD DEV   "
  FOR L = 1 TO NPERIODS WRITE VFLOW(K,L) AS D(8,3)
  WRITE AS /, "NUMBER"
  FOR L = 1 TO NPERIODS WRITE NFLOW(K,L) AS I 8
  WRITE AS /, /
LOOP  ''END OF FLOW RESULTS
```

```
  START NEW PAGE
  WRITE AS / , / ,B 5, "FLOW TIME STATISTICS FOR GROUP MEANS", /, /
  FOR K = 1 TO NCLASSES DO
    WRITE K AS S 10, "   JOB CLASS", I 3, /
    WRITE AS "PERIOD  "
    FOR L = 1 TO NPERIODS WRITE L AS I 8
    WRITE AS /
    WRITE AS "AVERAGE   "
    FOR L = 1 TO NPERIODS WRITE GRPFLOW(K,L) AS D(8,3)
    WRITE AS /, "STD ERROR*"
    FOR L = 1 TO NPERIODS DO
      IF NGRPFLOW(K,L) GT 0
        LET STDERR = VGRPFLOW(K,L) / SQRT.F(NGRPFLOW(K,L))
        WRITE STDERR AS D(8,3)
      ALWAYS
      IF NGRPFLOW(K,L) EQ 0 WRITE AS S 8
      ALWAYS
    LOOP
    WRITE AS /, "NUMBER"
    FOR L = 1 TO NPERIODS WRITE NGRPFLOW(K,L) AS I 8
    WRITE AS /, /
  LOOP  ''END OF GROUP FLOW RESULTS
  WRITE AS "* STD ERROR = STD DEV / SQRT(NUMBER)", /,
    "  USE AS ESTIMATE OF STD DEV OF FLOW TIME SAMPLE MEANS" , /

  RETURN
END  ''OF ENDSIM


ROUTINE TRACE  ''FOR DEBUGGING, SEE ASSIGNMENT TO BETWEEN.V IN MAIN
  WRITE TIME.V AS /, "TIME.V=", D(12,6), S 4, "DUMPING QUEUE", /
  CALL DUMPQUE
  SKIP 1 LINE
  GO TO 'RUN', 'START', 'SAMP', 'ARR', 'EPR' PER EVENT.V
'RUN' WRITE AS "RUNIN", /
  RETURN
'START' WRITE TRIALS AS "STARTDATA  TRIALS=", I 4, /
  RETURN
'SAMP' WRITE POINT AS "SAMPLE  POINT=", I 4, /
  RETURN
'ARR' WRITE CLASS, PERIOD AS "ARRIVAL  CLASS=", I 4, S 4, "PERIOD=",
        I 4, /
  RETURN
'EPR' WRITE ESERVER, EJOB AS "EPROC    ESERVER=", I 4, S 4, "EJOB=",
        I 10, /
  RETURN
END  '' OF TRACE
```

```
ROUTINE DUMPQUE
''TAKE OUT THE RETURN BELOW TO GET THIS TO DO ITS THING
 '' RETURN
  IF QUEUE IS EMPTY WRITE AS "QUEUE IS EMPTY", /
    RETURN
  OTHERWISE
  PRINT 1 LINE LIKE THIS
   JOB      ARRTIME      TYPE   PROCTIME    PRIORITY     ARRPRD
  FOR EACH J OF QUEUE
   PRINT 1 LINE WITH J, ARRTIME(J), TYPE(J), PROCTIME(J), PRIORITY(J),
      ARRPRD(J) LIKE THIS
**********  *****.*******    *    **.*******   *****.*********    *
  RETURN
END  ''OF DUMPQUE


ROUTINE GETINT YIELDING INTVAR
  DEFINE XREAL AS A REAL VARIABLE
    CALL FIND.EQUAL
    READ AS B BUFF.COLUMN USING THE BUFFER
    READ XREAL USING THE BUFFER
  ''DEBUG  WRITE XREAL AS /, "GETINT XREAL=", D(10,5)
    LET INTVAR=XREAL
  RETURN
END ''GETINT


ROUTINE GETREAL YIELDING XREAL
  DEFINE XREAL AS A REAL VARIABLE
    CALL FIND.EQUAL
    READ AS B BUFF.COLUMN USING THE BUFFER
    READ XREAL USING THE BUFFER
  ''DEBUG  WRITE XREAL AS /, "GETREAL XREAL=", D(10,5)
  RETURN
END ''GETREAL


ROUTINE FIND.EQUAL
 ''DEBUG  WRITE AS /,"ENTER FIND.EQUAL"
 ''SCANS INPUT BUFFER FOR A "=", AFTER WHICH AN INPUT VALUE
 ''SHOULD APPEAR.  ASSUMES FIXED INPUT RECORD OF 80 COLUMNS
    IF BUFF.COLUMN LT 0  CALL LOAD.BUFFER
    ALWAYS
  'SCAN' LET CHAR1=""
    FOR I=BUFF.COLUMN+1 TO 80, WHILE CHAR1 NE "=",
      READ CHAR1 AS B I, A 1 USING THE BUFFER
    IF I GE 80  CALL LOAD.BUFFER
      GO TO 'SCAN'
    ALWAYS
    LET BUFF.COLUMN=I
  ''DEBUG  WRITE BUFF.COLUMN AS /, "BUFF.COLUMN=", I 5
  RETURN
END ''FIND.EQUAL
```

```
ROUTINE LOAD.BUFFER
  DEFINE CHAR4 AS AN ALPHA VARIABLE
    START NEW INPUT LINE
    WRITE AS / USING THE BUFFER  ''CLEAR BUFFER
  ''DEBUG  WRITE AS /, "LOAD.BUFFER="
    FOR I=1 TO 20, DO
      READ CHAR4 AS A 4
   ''DEBUG   WRITE CHAR4 AS A 4
      WRITE CHAR4 AS A 4 USING THE BUFFER
    LOOP
    LET BUFF.COLUMN=0
  RETURN
END ''LOAD.BUFFER
```

BIBLIOGRAPHY

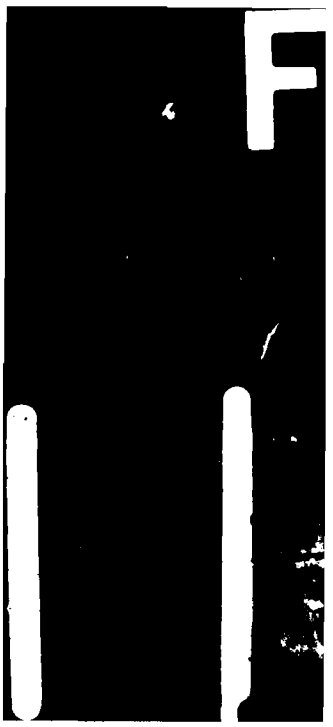
CACI, *SIMSCRIPT II.5 Reference Handbook,* Consolidated Analysis Centers Inc., Los Angeles, 1976.

Crawford, G. B., *Palm's Theoren for Nonstationary Processes,* The Rand Corporation, R-2750-RC, October 1981.

Hillestad, R. J., *Dyna-METRIC: Dynamic-Multi-Echelon Technique for Recoverable Item Control,* The Rand Corporation, R-2785-AF, July 1982.

Hillestad, R. J. and M. J. Carrillo, "Models and Techniques for Recoverable Item Stockage When Demands and the Repair Process are Nonstationary--Part I: Performance Measurement," The Rand Corporation, N-1482-AF, November 1980.

Kiviat, P. J., et al., *SIMSCRIPT II Programming Language,* Prentice-Hall, Englewood Cliffs, N.J, 1968.

Sherbrooke, Craig. C., *A Management Perspective on METRIC--Multi-Echelon Technique for Recoverable Item Control,* The Rand Corporation, RM-5078/1-PR, January 1968.

Takacs, L., *Introduction to the Theory of Queues,* Oxford University Press, New York, 1962.